NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

INCREASING OPEN SOURCE SOFTWARE INTEGRATION ON THE DEPARTMENT OF DEFENSE UNCLASSIFIED DESKTOP

by

Steven Anthony Schearer

June 2008

Thesis Advisor:	Rex Buddenberg
Second Reader:	Douglas E. Brinkley

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2008	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Increasing Open Source Software Integration on the Department of Defense Unclassified Desktop		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Steven A. Schearer			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

The United States Department of Defense (DoD) spends hundreds of millions of dollars each year on desktop computer software. While some of this expenditure goes to fund special-purpose military software, much of it is absorbed by license fees for computer operating systems and general-purpose office automation applications. Although many of these tools may serve their respective purposes rather well, there are many reasons to consider adopting alternative software solutions alongside the existing standards. Improvements to cost, security, and flexibility are some of the benefits that may be realized by integrating some of the many available mature, robust Open Source Software (OSS) solutions. In particular, Linux-based operating systems have helped bring free, open source software into mainstream use in businesses, homes, and government offices around the world, precisely because of these potential benefits. This thesis examines the feasibility of using OSS, particularly Linux-based operating systems, on unclassified DoD desktop computers. Specific attention is paid to performing office automation tasks that are currently handled by U.S. Air Force Secure Desktop Configuration, Windows-based computers. Additionally, this document examines many of the regulations and policies that shape the procurement and operational environments in which OSS must compete and function.

14. SUBJECT TERMS Open Source Software, OSS, Linux, Department of Defense, Unclassified, Desktop, Operating System, FOSS, GPL, PKI			**15. NUMBER OF PAGES** 89
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

INCREASING OPEN SOURCE SOFTWARE INTEGRATION ON THE DEPARTMENT OF DEFENSE UNCLASSIFIED DESKTOP

Steven A. Schearer
Captain, United States Air Force
B.A., University of Florida, 1999

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY
(COMMAND, CONTROL AND COMMUNICATIONS (C-3))**

from the

**NAVAL POSTGRADUATE SCHOOL
June 2008**

Author: Steven A. Schearer

Approved by: Rex Buddenberg
Thesis Advisor

Douglas Brinkley, EdD
Second Reader

Dan C. Boger, PhD
Chairman, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The United States Department of Defense (DoD) spends hundreds of millions of dollars each year on desktop computer software. While some of this expenditure goes to fund special-purpose military software, much of it is absorbed by license fees for computer operating systems and general purpose office automation applications. Although many of these tools may serve their respective purposes rather well, there are many reasons to consider adopting alternative software solutions alongside the existing standards. Improvements to cost, security, and flexibility are some of the benefits that may be realized by integrating some of the many available mature, robust Open Source Software (OSS) solutions. In particular, Linux-based operating systems have helped bring free, open source software into mainstream use in businesses, homes, and government offices around the world, precisely because of these potential benefits. This thesis examines the feasibility of using OSS, particularly Linux-based operating systems, on unclassified DoD desktop computers. Specific attention is paid to performing office automation tasks that are currently handled by U.S. Air Force Standard Desktop Configuration, Windows-based computers. Additionally, this document examines many of the regulations and policies that shape the procurement and operational environments in which OSS must compete and function.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This document was written entirely using the OpenOffice.org 2.3 word processor running on the Ubuntu 7.10 Linux-based operating system. Online research was conducted using the Firefox web browser, and research e-mails were exchanged primarily by means of the Evolution e-mail client. These last two applications were used (as required) in conjunction with the Common Access Card for interoperability with the Department of Defense Public Key Infrastructure. All of the software products mentioned are open source software. They all proved more than adequate for their respective tasks, and all of them were legally free to download, use, and (if so desired) modify. In the intervening months since this research project was started, each of these tools have also been actively maintained, released as updated versions with sometimes significantly enhanced features. I would like to thank the open source community; individuals, government agencies and corporations who helped develop this amazing array of software.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. OBJECTIVE

The United States Department of Defense spends hundreds of millions of dollars each year on desktop computer software. While some of this expenditure goes to fund special-purpose military software, much of it is absorbed by license fees for computer operating systems and general-purpose office automation applications. Although many of these tools may serve their respective purposes rather well, there is reason to consider adopting alternative software solutions alongside the existing standards. Improvements to cost, security and flexibility are some of the benefits that may be realized by integrating some of the many open source software solutions that now exist. In particular, Linux-based operating systems have helped bring free, open source software into the mainstream in businesses, homes and government offices around the world, precisely because of these potential benefits.

The Department of Defense (DoD) has made use of Linux-based systems in both the classified and server arenas, but there is room for increased use. The objective of this thesis is to examine the feasibility of incorporating Linux-based operating systems and other open source desktop application software alternatives on desktop computers across Department of Defense unclassified networks.

B. SCOPE

This thesis will begin with an examination of the current state of unclassified desktop software on Department of Defense computer systems, in part to determine what type of functionality and interoperability must be provided by any prospective open source solutions. The analysis will then focus on the integration of Linux-based systems into a Windows-dominated local area network. Server configurations are beyond the scope of the present paper,

which will focus solely on unclassified desktop technology. It is worth noting that many of the assembled facts and conclusions also apply to the DoD classified network environment.

C. RESEARCH QUESTIONS

- What, if any, are the potential advantages and disadvantages of introducing open source operating systems and other applications into the unclassified network environment?
- Can open source-based desktops integrate seamlessly into a network of primarily Windows-based servers and desktop clients?
- Can open source-based desktops accommodate the burgeoning Public Key Infrastructure, to include the Common Access Card token?
- How do federal government and Department of Defense regulations affect the procurement and use of open source software?

D. OUTLINE AND CHAPTER ORGANIZATION

This thesis is organized as follows:

Chapter II provides the reader with an overview of open source software and its applicability to the Department of Defense. Special attention is paid to the Linux kernel and derived operating system distributions. Next, the discussion moves to current unclassified desktop systems, with an emphasis on the standardized configuration used by the U.S. Air Force.

Chapter III continues with an examination of how open source software might be able to satisfy the needs of the user as they are currently met by the current U.S. Air Force proprietary software setup. A separate section is devoted to open source software interoperability with the Department of Defense Public Key Infrastructure and Common Access Card, due to their increasing significance. The chapter concludes with a review of open source software security.

Chapter IV begins with an overview of significant federal and Department of Defense information assurance policies that regulate the use of open source (and proprietary) software on official networks. The chapter continues with a look at some of the legal and technical challenges facing widespread Department of Defense adoption of open source software.

Finally, Chapter V briefly covers acquisition policy and offers recommendations to any Department of Defense officials who may be in a position to procure open source software for their respective agencies. It concludes with a brief list of topics that were perhaps beyond the scope of this thesis, but warrant further study.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. OVERVIEW

1. Open Source Software Introduction

General purpose computer workstation software can be categorized in many ways, such as by function, target architecture, or cost. One of the most fundamental and increasingly important distinctions lies in the accessibility to and licensing of a program's source code. This code is the human-readable, though perhaps arcane-looking blueprint of a program's design. With the source code, a programmer can review the underlying design of a program and make any conceivable change to that program's functionality. By means of a compiler, the programmer can then transform the code into a binary format that only the computer can interpret. At that point, the program can be run as a potentially useful tool like a web browser or word processor, but it can no longer be modified, and its inner workings cannot be examined. Therefore, free and legal access to the source code grants the user or operating agency considerable power and control. Commercial, off-the-shelf (COTS) software for which source code is freely available is known as Open Source Software (OSS). The Open Source Initiative (http://www.opensource.org) further refines the OSS definition to include free redistribution of source code and permission to modify and redistribute derivative works, among other things.

Conversely, the Department of the Navy (DON) Chief Information Officer established in 2007 that "commands will treat OSS as COTS when it meets the definition of a commercial item . . . This will allow the DON to utilize OSS throughout the enterprise when acquiring capabilities to meet DON business and warfighter requirements" (U.S. DON CIO, 2007). Due to the public availability of source code, mature OSS products are often the result of considerable public collaboration. And since most contributors (including those from numerous

corporations such as IBM, Oracle and Google) would object to having their work surreptitiously bundled up and sealed into another vendor's closed-source product, OSS is typically distributed with one of several licenses that demand re-release of any modified source whenever modified binary files are published. Partly because of these licenses, the software is usually (though not always) available free of charge, if only in a non-compiled, source-only format. Clearly, the up-front cost savings can be significant if a software package is available for free. However, many popular OSS operating systems and applications are sold in a more usable, pre-compiled format by companies who tailor their products to meet certain needs and hope to generate sales largely on the basis of continuing support. One example is the Red Hat Corporation, which sells Red Hat Enterprise Linux in both desktop and various server configurations. The source code to these products is still available free of charge.

OSS such as Red Hat Enterprise Linux is already in use throughout the Department of Defense (DoD) in everything from embedded vehicle computers to highly relied-upon directory and web servers. In their 2003 report commissioned by the Defense Information Systems Agency, MITRE came to the conclusion that "FOSS software plays a far more critical role in the DoD than has been generally recognized. The value of FOSS to the DoD appears to be greatest in four broad categories: Infrastructure Support, Software Development, Security, and Research" (MITRE, 2003, p. 17). In fact, it will likely serve as the basis for the U.S. Army's Future Combat Systems (Klein, 2008). It has, however, failed to make inroads in the general-purpose DoD desktop computing environment. Possible reasons for this are many and varied, but the most visible and significant barrier to entry is undoubtedly the dominant global market share of the Microsoft Windows operating systems and accompanying Office productivity suite. Having established a massive user base with applications, file formats and network protocols that are exclusive to the Windows/Office environment, Microsoft has developed the de facto standard for desktop computing.

Those who would stray from this path risk sacrificing interoperability, perhaps even with their organization's own legacy programs and data files. However, considerable progress has been made in recent years to allow OSS programs to work with Microsoft Office files and interact with Windows-based computers. Full Application Programming Interface (API) compatibility remains the major stumbling block, but with the advent of web-based computing, many applications are now web-dependent instead of OS-dependent (Boutin, 2006). These facts, along with OSS offerings that rival their closed-source counterparts, are making OSS an increasingly attractive alternative to businesses and governments around the world (CSIS, 2007).

By the very nature of its name, Open Source Software brings perhaps its biggest benefit to the table: peer review. The quality benefits imparted by code review cannot be overstated. By some estimates, inspections eliminate 70 to 80 percent of all software defects (Petross, MN3331 lecture, Winter 2008). In his book *Managing the Software Process*, Watts Humphrey cites many noteworthy examples of the great successes companies have had in examining code, including the following: "In an AT&T Bell Laboratories dial-up central switching system application, inspections were reported to be 20 times more effective than testing in finding errors" (Humphrey, 1989, p. 185).

2. Open Source Software Initial Acquisition Costs

As previously mentioned, not all OSS is intended to be used for free. The best example of this is Red Hat Enterprise Linux (RHEL). Since it is an operating system largely derived from other publicly developed OSS, it is subject to several popular OSS licenses, including re-release of modified source code. This is demanded in particular by the GNU (a recursive acronym for GNU's Not Unix) General Public License, perhaps the most popular OSS license in use. While Red Hat is under no obligation to release free binary, immediately usable versions of its product, it must make the source code available. Other entities can (and do) then take this code, re-compile it into binary form, and legally walk

away with a nearly identical, fully functional binary product. The best-known example of this is CentOS, a product that is in almost every way a clone of RHEL, with Red Hat's trademarked logos removed.

To create and protect a revenue stream, therefore, Red Hat requires that users of RHEL pay a subscription fee in order to download updates and additional programs from the Red Hat Network. This fee also entitles users to unlimited web support with a two-business-day turnaround time. The retail price for a one year, basic subscription to this service is $80, or three years for $228. “Security errata and select mission-critical bug fixes” are available for seven years from the general availability of the product. By comparison, as of the writing of this paper, the latest version of Windows Vista Ultimate costs between $300 and $400 through most major web retailers. It comes with 90 days of telephone support, after which users must pay a $59 per-call fee. Bug fixes and other software updates are provided by Microsoft for a period of five years after product release. The retail prices listed here would no doubt be significantly lower in the case of a bulk contract purchase. And while the OSS offering in this case is cheaper than the proprietary alternative, these figures are used primarily to illustrate the point that OSS can carry a significant up-front financial cost. On the other hand, it is worth pointing out that RHEL is not an operating system alone; it is delivered with a complete office suite (OpenOffice.org) and hundreds of other programs that can optionally be installed from the DVD media or through the Red Hat Network.

3. Vendor Lock-in

One major factor that exacerbates the cost dimension of proprietary software is vendor lock-in, which occurs when a customer becomes dependent on a piece of software (or hardware) to the extent that switching vendors would present significant challenges and costs. Much of vendor lock-in is dependent on whether or not the software uses open communications protocols and document format specifications, but it can also apply to the underlying API’s. In 2004, the

European Union (EU) Commission released a decision regarding Microsoft's antitrust actions in the EU. In it, they referenced an internal memo prepared for Bill Gates from Microsoft C++ General Manger Aaron Contorer (European Commission, 2004, p. 127):

> The Windows API is so broad, so deep, and so functional that most ISV's [Independent Software Vendors] would be crazy not to use it. And it is so deeply embedded in the source code of many Windows apps that there is a huge switching cost to using a different operating system instead . . . . It is this switching cost that has given customers the patience to stick with Windows through all our mistakes, our buggy drivers, our high TCO [Total Cost of Ownership], our lack of a sexy vision at times, and many other difficulties . . . . Customers constantly evaluate other desktop platforms, [but] it would be so much work to move over that they hope we just improve Windows rather than force them to move. In short, without this exclusive franchise called the Windows API, we would have been dead a long time ago . . .

In the same document, Microsoft Senior Vice President Bob Muglia is quoted as saying "The Windows franchise is fueled by application development which is focused on our core API's" (European Commission, 2004, p. 127).

The realities behind these powerful statements have been felt and understood by users, managers, developers and resellers throughout the computing world for almost two decades. Those ISV's that want to earn maximum profit from their coding efforts typically target the Windows API and corresponding user base. This development results in a large pool of available Windows applications. Correspondingly, users who want maximum commercial software choice tend to purchase Windows-based systems, creating a large Windows user base, and so on and so forth.

Operating system-specific application development creates this interesting chicken-and-egg scenario for supply and demand, but there should be no such inherent dilemma for disparate interconnected computer systems. However, when organizations choose proprietary software solutions with closed communication protocols, vendor lock-in occurs here, too. For example, many information systems managers understand that if they wish to use Microsoft

Outlook in its native Mail API mode for corporate e-mail, calendar, task and other groupware functions, then they must continue to use Microsoft Exchange as the groupware server application. And if they wish to use Exchange, they must run it on a Microsoft Windows-based server. Likewise, if they want the Windows server to operate seamlessly with other corporate servers such as the directory and client management system, then it must all be tied together with Microsoft Active Directory. (These relationships tend to hold true in reverse, too). This inability to incorporate new, competing technologies from different vendors presents an interesting, if unfortunate cost variable far beyond the simple price point of the software. Without a massive investment in infrastructure across the board, the organization is effectively locked-in to one vendor's solutions. The U.S. Navy's Open Architecture Contract Guidebook addresses this in its Life Cycle Affordability checklist: "Have proprietary products been avoided to avoid vendor lock-in and sole source environments?" (PEO-IWS, 2006 p. 56).

The U.S. Army seems to have recognized the potential for vendor lock-in and has addressed it in the Future Combat Systems (FCS) program. In particular, the operating system software to be used across the FCS, known as they System of Systems Common Operating Environment (SYSCOE) is being developed by Boeing and will be based on Linux. According to the Washington Post, "Boeing and the Army said they chose not to use Microsoft's proprietary software because they didn't want to be beholden to the company. Instead, they chose to develop a Linux-based operating system based on publicly available code" (Klein, 2008). Although the entire SYSCOE platform will undoubtedly be a complex product, the OSS-based architecture may level the playing field for future support contracts; any number of Linux-savvy vendors could theoretically step in to assume programming and support roles.

4. About the Linux Kernel

Any casual discussion about OSS will likely include mention of Linux, as it is perhaps the most prominent piece of Free, Open Source Software in the world.

In everyday language, Linux has come to be known as a complete operating system, to include graphical user interface features and rich multimedia capabilities. However, the word Linux officially refers only to the operating system kernel; that core piece of software that controls system hardware and allows the computer to interpret human language commands, among other things. DoD information technology managers unfamiliar with Linux are often curious about who exactly owns and maintains this piece of technology upon which we already rely so heavily. The Linux kernel is a patchwork of contributions by thousands of individuals, some working as volunteers and others in the employ of companies which may stand to benefit from improvement of the operating system. In April 2008, the Linux Foundation published a study that cataloged contributions to the 2.6.x Linux kernel according to a series of metrics. The paper's authors attempted to track down the employers of Linux kernel developers in an effort to shed light on which companies were behind some of the kernel's growth. Some of this research is summarized in Table 1 below. The following excerpt clarifies the findings in the table:

> There are a number of developers for whom we were unable to determine a corporate affiliation; those are grouped under "unknown" . . . . With few exceptions, all of the people in this category have contributed 10 or fewer changes to the kernel over the past three years, yet the large number of these developers causes their total contribution to be quite high. The category "None," instead, represents developers who are known to be doing this work on their own, with no financial contribution happening from any company. The top 10 contributors, including the groups "unknown" and "none" make up over 75% of the total contributions to the kernel. It is worth noting that, even if one assumes that all of the "unknown" contributors were working on their own time, over 70% of all kernel development is demonstrably done by developers who are being paid for their work.

Company Name	# of Changes	% of Total
None	11,594	13.9%
Unknown	10,803	12.9%
Red Hat	9,351	11.2%
Novell	7,385	8.9%
IBM	6,952	8.3%
Intel	3,388	4.1%
Linux Foundation	2,160	2.6%
Consultant	2,055	2.5%
SGI	1,649	2.0%
MIPS Technologies	1,341	1.6%
Oracle	1,122	1.3%
MontaVista	1,010	1.2%
Google	965	1.1%
Linutronix	817	1.0%
HP	765	0.9%
NetApp	764	0.9%
SWsoft	762	0.9%
Renesas Technology	759	0.9%
Freescale	730	0.9%
Astaro	715	0.9%
Academia	656	0.8%
Cisco	442	0.5%
Simtec	437	0.5%
Linux Networx	434	0.5%
QLogic	398	0.5%
Fujitsu	389	0.5%
Broadcom	385	0.5%
Analog Devices	358	0.4%
Mandriva	329	0.4%
Mellanox	294	0.4%
Snapgear	285	0.3%

Table 1. Kernel Lines of Code Changed, by Employer
Source: Corbet et al., 2008

This 70% of paid kernel development is done not out of charity. Rather, the backing corporations understand that OSS can be profitable. A more capable and robust Linux kernel, devoid of licensing fees and royalties, makes an extremely powerful and cost-effective foundation on which to develop an ever-growing number of information systems. Linux is used in everything from embedded systems and network appliances to enterprise servers and some of the world's most powerful multi-processor supercomputers.

Since the kernel updates are released on a schedule and contributions are checked or “signed off” by someone before public release, there must be some overarching authority maintaining responsibility and a degree of ownership. In the purest sense, that authority is Linus Torvalds, the original kernel's creator. In his book *Open Life – The Philosophy of Open Source*, Henrik Ingo colorfully explains the Torvalds' role as one of a “benevolent dictator,” albeit one with strangely limited power:

> What would happen if for some reason Linus decided to screw things up and out of spite started making stupid decisions for Linux? Within twenty-four hours the other Linux developers would leave him to fool around on his own, make a copy of the Linux source code somewhere Linus couldn’t get his hands on it and keep working without him. It’s also extremely likely that the hackers involved would quickly elect – more or less consciously and more or less democratically – a new benevolent dictator.
>
> All that is possible because the code itself is open and freely available for anyone to use. As dictator, Linus has all the authority while at the same time having no power whatsoever. The others see him as their leader only because he is so talented – or benevolent. There is a fascinating equilibrium of power and freedom. The dictator has the power and the others have the freedom to vote with their feet.

While Linus Torvalds retains this “ownership” of the kernel, in practice, and due to the volume of code in question, other individuals have assumed responsibility for different branches. According to Ingo (p. 45):

> Linus in particular takes the advice of his closest and longer-term colleagues, who within the community are known as his lieutenants. These lieutenants are like mini-dictators, and each one has their own area of responsibility within the project. Just as for Linus, their authority is based on talent proven over a period of years and the trust that it has generated. The dictatorship is therefore a meritocracy.

5. Linux Distributions

Because the word “Linux” has differing meanings to different people, the term “Linux distribution” is used to bridge the gap between the stand-alone kernel and an entire packaged, fully functional Linux desktop or server operating system. As of early 2008, the Linux advocacy site http://www.linux.org listed 220 actively maintained distributions.

While Linux kernel releases remain the domain of Linus Torvalds, distributions can be created and maintained by anyone. Each Linux distribution, while maintaining some basic level of commonality with other distributions (if only due to kernel pedigree), brings something unique to its target user audience. For example, Damn Small Linux (DSL) was stripped of a multitude of features found in most distributions in favor of delivering a very lightweight operating system with minimal system memory requirements. It runs acceptably on old hardware and boots from removable media such as CDs and USB memory sticks. The Backtrack distribution was created with computer system forensics in mind. Ubuntu, sponsored by the Canonical Corporation, has placed its emphasis on user-friendliness. As a result, it has gained a significant user base over the past few years, rising to become the most popular distribution (in the last six months) on http://distrowatch.com, a popular Linux distribution-tracking website. Finally, Red Hat Enterprise Linux, a fee-based product, has targeted the government and corporate markets, offering sophisticated directory server capabilities, a strict software testing and release methodology, and other features prized by organizations with a large number of managed client systems.

B. CURRENT UNCLASSIFIED DESKTOP SYSTEMS

1. Consolidated Acquisitions

The Department of Defense has not mandated exclusive use of the Microsoft Windows operating system for unclassified desktop computing. Rather, following the civilian marketplace, the Services' networks have evolved to

use Microsoft's software as their primary unclassified desktop environments. Since the early part of this century, the Services have also taken it upon themselves to formally standardize their purchasing and configuration of Microsoft software. This was done in part to reduce the costs associated with multiple smaller buys (leveraging volume licensing agreements) and inadvertent redundant license purchases across the enterprise. The U.S. Army led the charge in Service-wide Microsoft Enterprise Licensing Agreements (ELAs) in 2003, saying “the deal would save millions of dollars in operational costs and improve software license and asset management” (Wait, 2003). Through its own recent 500 million dollar, six-year Microsoft ELA, the U.S. Air Force consolidated 38 separate purchase agreements and projected expected savings of over 100 million dollars. In both cases, however, security was at least as important as cost in the ELA decision. In fact, John Gilligan, U.S. Air Force Chief Information Officer at the time that Service's ELA was established, stated that “The major driver was probably security.” The reasoning behind Gilligan's assertion is described in the following section.

2. Standardized Configurations

In 2006, to reap the security benefits of its new ELA, the U.S. Air Force worked directly with Microsoft to deploy a standardized version of the Windows XP desktop operating system, dubbed the Standard Desktop Configuration (SDC). By means of very deliberate configuration control, this effort was “part of an overall objective to increase security and reduce lifecycle management costs associated with desktop computer systems” (Yasin, 2007). The SDC includes not only the operating system, but also other elements considered essential to general productivity, systems management and security in the Air Force enterprise network (see Table 2). At the end of 2007, the SDC was being upgraded to the Windows Vista operating system and the Office 2007 productivity suite.

Application	Manufacturer	Version
Windows XP SP2 with Firewall Enabled	Microsoft	SP2
SMS 2003 Client	Microsoft	2003
Norton Anti-Virus	Symantec	10.0.2
.NET Framework SP2	Microsoft	1.1
ActivCard Gold Card Reader Software	Activcard	3.0 FP1 USAF
Office 2003	Microsoft	2003
Visio 2003 Viewer	Microsoft	2003
ICS Viewer	PureEdge	6.0.1
MasterKey Plus (for DMS)	Boldon James	4.2.2
DoD Banner w/Screen Saver	DoD	3.1
Internet Explorer	Microsoft	6.0.2900.2810
Acrobat Reader	Adobe	7.0.7
Quicktime Player	Apple	7.0.4
Windows Media Player	Microsoft	10
Java Runtime Engine	Sun	1.5
Macromedia Flash Player Plug-in	Macromedia	Latest
Shockwave Player Plug-In	Macromedia	Latest
MDAC	Microsoft	Latest
DoD PKI Certificates	DoD	

Table 2. U.S. Air Force SDC, November, 2007
Source: U.S. Air Force 754th Electronic Systems Group

The U.S. Air Force is not alone in its standardization efforts. The Army produces a "Gold Master" and the U.S. Navy has its own "Workstation Baseline Software Configuration Gold Disk," both of which are Service-specific versions of the SDC. At the federal level, "The Office of Management and Budget requires all agencies to migrate to a standard desktop configuration for Microsoft Windows XP and Vista environments by February 2008" (Yasin, 2007). These standardization efforts, while focused on the Microsoft Windows platform, do not demand Service-wide use of Microsoft products; they simply mandate a certain configuration when Microsoft products are used. Similar security-based standard

configuration guidelines are also dictated by the Defense Information Systems Agency (DISA) for any Unix-like platforms, to include Linux.

All of the standardization taking place in the Microsoft-based DoD networks actually simplifies the task of integrating Linux-based systems. First, it provides a solid system architecture baseline with which any foreign hosts must interact. Second, it establishes capability baselines; minimal functionality specifications which must be met in order for the new systems to be considered adequate peers.

THIS PAGE INTENTIONALLY LEFT BLANK

III. OPEN SOURCE SOFTWARE OPTIONS AND CAPABILITIES

A. FUNCTIONAL COMPARISON

1. Operating System

The U.S. Air Force Standard Desktop Configuration offers a good starting point from which to evaluate potential OSS alternatives. At the top of the list in Table 2, the operating system establishes the foundation for a comprehensive system configuration. While there are several noteworthy OSS operating systems, including the mature Free BSD and Minix operating systems, DoD agencies are restricted by instructions and regulations as to which operating systems may be used on official, production networks. This topic is addressed further in Chapter V of this document.

In short, as described by the U.S. Air Force Communications Agency in a telephone interview on April 25 2008, operating systems are only considered for use if they have been certified at Evaluated Assurance Level (EAL) 4+ in the National Information Assurance Partnership's (NIAP) Common Criteria Evaluation and Validation Scheme (CCEVS). Additionally, certification testing for EAL 4+ must have taken place in a U.S. lab, per DoD Instruction 8500.2 and the NIAP's requirements for Common Criteria Testing Laboratories. In the OSS world, this limits options to two vendors' Linux distributions. As of the writing of this document, only Red Hat Enterprise Linux (various versions 4 and 5) and SUSE Linux Enterprise Server (version 10) had Common Criteria EAL 4+ from U.S. labs, according to the NIAP-CCEVS Validated Products List. In the past year, Sun Microsystems has made strides in opening the source to its Solaris operating system, and version 10 has been awarded EAL 4+ (albeit from a Canadian lab). However, the OpenSolaris project is not entirely the same as the Solaris operating system. According to the http://OpenSolaris.org website, it is “an open source project sponsored by Sun Microsystems, Inc, that is initially

based on a subset of the source code for the Solaris Operating System... [It] will find a variety of uses, including being the basis for future versions of the Solaris OS product, other operating system projects, and third-party products and distributions."

Primarily because of accessibility (in the form of a Naval Postgraduate School site license), the Red Hat Enterprise Linux 5 Desktop distribution was informally evaluated for the purposes of this study. In particular, it had to meet the author's needs as a general-purpose workstation, capable of using the same network resources and performing the same tasks as a similar Windows XP workstation. In relation to these goals, the RHEL system performed well. With its wired Ethernet connection to the Naval Postgraduate School's local area network, the system provided easy access to shared drives, intranet websites and printers. Any necessary configuration was accomplished entirely in the point-and-click graphical user interface.

2. Enterprise Management

As noted next on the SDC list, Windows clients on many military installations are equipped with the Microsoft Systems Management Server (SMS) client-side application. This application interfaces with an SMS server to allow for remote control, inventory, and other types of configuration management. While this particular piece of software is unique to the Microsoft Windows environment, RHEL offers similar tools that would be highly useful in remotely managing RHEL desktop systems. The Red Hat Network web-based management system provides inventory capabilities for all registered systems and allows administrators to remotely deploy or remove software packages to and from specified groups of systems. Enterprise-wide system management, while somewhat beyond the scope of this document, cannot be overlooked if a large number of Linux-based clients are to be introduced into an otherwise Windows-dominated network.

3. Anti-Virus

Since its inception, the Linux kernel has been far less affected by computer malware than the various Windows releases. Whether or not this stems largely from the fact that the operating system is significantly less targeted by malware authors is a matter of debate. The other side of this argument typically states that the massive peer review enjoyed by OSS gives it a much cleaner, robust code base, as well as a faster and more transparent correction mechanism for any discovered flaws. According to one recent estimate, out of over 236,000 malware items, “only about 700 are meant for the various Unix/Linux distributions” (van Oers, 2007).

While many Linux users do not employ any anti-virus software whatsoever, DoD regulations stipulate otherwise. The Defense Information Systems Agency (DISA) is responsible for establishing such policy Department-wide (DISA STIG, 2006):

> Department of Defense Directive (DODD) 8500.1 establishes policy and assigns responsibilities to the Defense Information Systems Agency (DISA) to develop and provide security configuration guidance for IA and IA-enabled IT (Information Technology) products in coordination with the National Security Agency (NSA). Paragraph 4.18 of the 8500.1 states, "All IA and IA-enabled IT products incorporated into DOD information systems shall be configured in accordance with DOD-approved security configuration guidelines." DISA Field Security Operations (FSO) develops the guidelines, which are called Security Technical Implementation Guides. . . . . any UNIX based operating system in use in a DOD environment is subject to all relevant UNIX security requirements and must be capable of STIG compliance as verified by a Security Readiness Review (SRR).

According to DISA's Unix STIG (a general document for Unix-related systems which also specifically applies to Linux) Security Readiness Review (SRR) Checklist, anti-virus software must be installed and set to scan the entire system automatically at weekly intervals. While the U.S. Air Force SDC uses Norton (Symantec) Anti-Virus Corporate Edition, the STIG SRR refers to the McAffee command-line tool for Unix, and demands “an approved DoD virus scan

program” (DISA Unix SRR, p. 269). Whether it is used to defend against the few known threats, or to prevent Windows malware from being propagated via the Linux systems, anti-virus software is a requirement for DoD Linux clients.

4. Smart Card Middleware

One increasingly important distinguishing feature of the RHEL 5 distribution is its fully functional, out-of-the-box smart card support for many card readers. Smart cards such as the DoD's Common Access Card (CAC) provide dual-factor authentication and encryption capabilities for more secure network log-ins and e-mail. In addition to the usual open source command-line tools, Red Hat provides its own “Smart Card Manager” graphical system panel applet. The combination of these programs negates the need for (and expense of) additional middleware, such as the ActivCard Gold software used on the Windows XP workstation. This topic is covered more extensively in section B of this chapter, is it is quickly becoming essential to DoD computing.

5. Productivity Suites

With regards to office automation, OpenOffice.org (the name of an application, not just its website) has emerged as the predominant open source office productivity suite. OpenOffice.org (known as OOo) is free of charge and comes with a word processor, as well as spreadsheet, presentation, schematic drawing and database programs. While OOo has made great progress in terms of being able to read from and write to Microsoft Office file formats, the closed (and constantly changing) nature of those binary file formats has historically made interoperability a challenging prospect for OOo programmers. Making the process easier was Microsoft's release of specifications on the Word, PowerPoint and Excel binary file formats in February of 2008 (http://www.microsoft.com/interop/docs/officebinaryformats.mspx). Currently, the OOo 2.4 word processor “Writer” can read from and write to Microsoft Office Word files from versions 6.0 to 2007. In fact, Writer was used exclusively to compose and edit this document, which was originally based on a Microsoft

Word 2003 template. Writer also has the built-in ability to export documents in the ubiquitous Portable Document Format (PDF); another requirement for final submission of this thesis.

Weighing against OOo is its lack of full compatibility with Microsoft Office files, as that suite is the de facto standard throughout not only the DoD, but much of the business world. While importing Office files is usually quite successful, irregularities and discrepancies sometimes surface. OOo Draw, a Visio-like program, cannot open or save to Microsoft Visio files at all, and Word/Excel users who wish to use the OOo Writer/Calc programs will not be able to utilize any existing Word/Excel macros. The Visio-related shortcomings of OOo are primarily related to importing and exporting native Visio files. One work-around is to save Visio files in the Visio-XML format instead. The OSS application Dia, by means of a plug-in, can then import, manipulate and export the files. Alternatively, users can switch to the more openly modifiable scalable vector graphics (SVG) format, which can be manipulated by a number of OSS applications, including Dia and OOo Draw.

Although these limitations will probably be of limited impact to the average office staff, they will undoubtedly cause some level of frustration among "power users." To further assist in making a comprehensive comparison, Idealware, a nonprofit software review organization, has published a fairly thorough feature review comparing Microsoft Office and OOo at http://www.idealware.org/articles/msoffice_vs_openoffice.php. In addition to the features previously mentioned, Idealware notes that OOo Writer does not contain a grammar checker, whereas Word 2003 does. A comparatively rudimentary checker called LanguageTool is available for Writer as a user-installable, third-party extension.

The OpenOffice.org developers do not pride themselves exclusively on compatibility with Microsoft file formats, and this cannot be the sole measure of the quality of the suite. In particular, OOo developers and users are generally pleased to report that they have the native ability to work with Open Document

(ODF) files; an International Standards Organization approved, royalty and license-free format. According to http://opendocument.xml.org/overview:

> From a technical point of view, ODF is a ZIP archive that contains a collection of different XML files as well as binary files like embedded images. The use of XML makes accessing the document content simple because content can be opened and changed with simple text editors if necessary. In contrast, the previously used binary file formats were cryptic and difficult to process. The ZIP compression guarantees relatively small file sizes, in order to reduce file storage and transmission bandwidth requirements.

As of late spring 2008, the average price of the Microsoft Office 2003 Professional suite was $425 for a single-user retail license. Office is not natively available for use on any Linux-based platform. Conversely, the OpenOffice.org suite is available as a free download for the Windows, Linux and Mac OSX operating systems, and is included with many Linux distributions, including Red Hat Enterprise Linux. As with all other enterprise software, a decision to employ a certain office productivity suite cannot be based solely on initial acquisition price, no matter how low. Product capabilities, as well as the significant other OSS advantages and challenges listed throughout this document, must factor into the decision.

The eighth item on the SDC list is the *ICS Viewer*. Both the U.S. Army and U.S. Air Force have used various incarnations of this software to edit electronic forms instead of paper records; thousands of forms have been converted to its Extensible Markup Language (XML) format. This process is vital to meeting goals of the paperless office initiatives mandated by the Government Paperwork Elimination Act of 1998. The PureEdge application itself has gone through several substantial changes in the past few years. Most significantly, the developer, PureEdge, was acquired by IBM in 2005, and the product was renamed *IBM Workplace Forms*. More recently, IBM transitioned the application into its Lotus suite, and it is now known as *IBM Lotus Forms*.

Across much of the U.S. Army and in parts of the U.S. Air Force, digital signature software by vendor Silanis is used in conjunction with the PureEdge and/or IBM form software to enable Common Access Card-based signatures, further reducing the need for paperwork. This process is, in most cases, entirely dependent on Windows-specific client-side software. However, the capability now exists to host the electronic forms on a web server and allow users to edit them in a web browser. According to an IBM press release about *Lotus Forms 3.0*, “Businesses and government organizations may now deliver core business processes to their customers or constituents via the Internet and enable forms completion — including digital signature capabilities — without concern for managing the software levels on the consumer's computer” (IBM, 2007). This capability may bring operating system independence to yet one more application, and enable digital forms on the Linux desktop.

6. E-mail

While Microsoft Office includes the Outlook e-mail and groupware client, OpenOffice.org does not provide any similar functionality, perhaps because several other independent (and already popular) products exist in the OSS community. Replicating Outlook-based e-mail functionality, however, may be the biggest obstacle to integrating general-purpose Linux workstations into any DoD Windows-dominated network. Most military installations rely on Microsoft Exchange for e-mail and groupware services. While an Exchange server is capable of providing mail services to clients via the industry-standard Post Office Protocol (POP) or Internet Message Access Protocol (IMAP), these capabilities have historically been disabled for security and supportability purposes. In addition, their use robs clients of most of the groupware functionality offered by the Exchange server, such as shared calendars, tasks, and other collaborative features. Therefore, most military installations only permit e-mail clients to connect to Exchange servers via Exchange's native, proprietary Message Application Programming Interface (MAPI).

Without the use of third-party middleware, no OSS e-mail client can connect to an Exchange server in MAPI mode. However, middleware known as Brutus (http://www.42tools.com) now exists to enable Novell Evolution e-mail client (and potentially others) to take full advantage of Exchange functionality. While employing this middleware server would theoretically be independent of the operation of the Exchange server, it still goes beyond simple addition of OSS-based desktop software and is therefore beyond the scope of this document. Currently, the simplest remedy to the OSS-Exchange barrier is the use of the Evolution e-mail client, with Outlook Web Access (OWA) enabled on the Exchange server. OWA need not be permitted to communicate outside the intranet (something that is often blocked due to security concerns); it simply serves as the communications channel for internal Evolution clients. When operated in "Exchange mode" (using OWA), the Evolution client provides both a look-and-feel and functionality that is very similar to that of Outlook. It can access the Exchange Global Address List and integrate with DoD Public Key Infrastructure, allowing users to send and receive digitally signed and encrypted e-mails (see Appendix A).

In addition to the closed communications protocol, Outlook and Exchange also employ a proprietary "personal folders" file format (.pst) that is not usable by OSS e-mail clients. This fact makes it difficult for users to fully access their email accounts, particularly stored e-mail, from disparate client platforms. The problem is not unique to OSS e-mail clients. Many network users in the U.S. Central Command (USCENTCOM) headquarters use Blackberry communication devices, which also cannot access .pst files. The solution at USCENTCOM is to avoid the use of .pst files altogether. Rather than giving users a large network storage home drive on which to store their .pst's, administrators allocate the space directly to Exchange mailboxes. This technique can also allow Evolution and other OSS e-mail client users to access all their mail from anywhere on the corporate local area network.

The final challenge facing OSS e-mail clients is the Defense Message System (DMS), a DoD-wide system of record for official message traffic, and the replacement for the Automatic Digital Network (AUTODIN). DMS is based on x.509 certificates and hardware tokens called Fortezza cards, but was designed exclusively around a Microsoft Exchange/Outlook architecture and a specialized directory tree. None of the required extension software is available for OSS e-mail clients or the Linux environment. Fortunately, DMS is not required for use by all DoD members, and sees the most use in classified networks. Additionally, according to the Defense Information Systems Agency's website, "DMS will continue to shift from a predominantly writer-to-reader topology to a domain Fortezza topology" (http://www.disa.mil/main/prodsol/dms.html). Software known as the Collaborative Messaging System (CMS) by Lockheed Martin, Boldon James and Microsoft, already exists to address this shift. It allows DMS access via a web browser, with hardware tokens being located at central server, a solution that may extend DMS functionality to non-Windows clients.

7. Other Third-party Viewers

The U.S. Air Force SDC includes the ability to play Apple Quicktime, Adobe Flash and Windows Media files and streams. This functionality is freely available on multiple Linux distributions, usually in the form of both open and closed-source solutions. The Java Runtime Engine (JRE) is available for Linux from both Sun and IBM. Acrobat Reader is available from Adobe, and several other OSS programs including OpenOffice.org allow for reading from and writing to PDF files.

8. Substitutions/Equivalents Overview

As described in the last seven sections, OSS offers an array of products to address every basic need of typical office automation computing. An overview of the potential alternatives to SDC applications can be found in Table 3 below.

Current SDC Software	OSS Equivalent/Substitution
Windows XP SP2 with Firewall Enabled	Red Hat Enterprise Linux 5.x
SMS 2003 Client	N/A
Norton Anti-Virus	McAffee Anti-Virus, Clam, etc.
.NET Framework SP2	N/A
ActivCard Gold Card Reader Software	N/A
Office 2003	OpenOffice.org 2.x
Visio 2003 Viewer	Dia
ICS Viewer	Possible Wine implementation
MasterKey Plus (for DMS)	N/A
DoD Banner w/Screen Saver	DoD Banner w/Screen Saver
Internet Explorer	Firefox
Acrobat Reader	Acrobat Reader or OSS variant
Quicktime Player	Totem or other OSS equivalent
Windows Media Player	Totem or other OSS equivalent
Java Runtime Engine	Java Runtime Engine
Macromedia Flash Player Plug-in	Adobe Flash Plug-in
Shockwave Player Plug-In	Adobe Flash Plug-in
MDAC	N/A
DoD PKI Certificates	DoD PKI Certificates

Table 3. List of SDC/OSS Equivalents/Substitutions

Some of the items listed in Table 3 are not available in the default installation of RHEL. However, RHEL and most other major Linux distributions provide access to upgrades, bug fixes and additional software packages through an online repository system. Systems can be set up to either automatically fetch and install updates, or users can retrieve additions manually via a search function. This capability can be exploited to address specific Service software requirements. Software installation tools such as yum can be pointed at several different repositories simultaneously. A Service would simply need to establish its own internal repository servers, populated with all required applications (either

proprietary or other perhaps modified/vetted OSS) to simplify installation and maintenance of applications that are not included on standard distributions.

B. PUBLIC KEY INFRASTRUCTURE AND COMMON ACCESS CARD

1. Overview and Regulations

The Common Access Card (CAC), with its embedded Public Key Infrastructure (PKI) certificates, is quickly becoming the primary authentication instrument for U.S. Department of Defense (DoD) information systems. While some internal LAN systems may still be accessed via username and password, initial login to the NIPRNet (Unclassified but Sensitive Internet Protocol [IP] Router Network) must now be accomplished via two-factor authentication. This radical security shift became effective across the DoD as of August 2006, according to Joint Task Force – Global Network Operations (JTF-GNO) Computer Tasking Order (CTO) 06-02. Unless otherwise approved by JTF-GNO, the required “two-factors” are the CAC and Personal Identification Number (PIN). Although some Internet-facing DoD web servers may (and do) still operate with username and password combinations, this practice is fading. Web resources such as the Air Force Portal and Army Knowledge Online accept the CAC and PIN as their primary means of authentication.

The CAC PKI system is also perfectly suited to email use, and almost all CAC-holders have the option to send and receive signed and/or encrypted email messages to and from other CAC users in the DoD PKI sphere. However, all of this functionality demands that the users' computer system be able to interface with the CAC, access the embedded key material and use the proper associated cryptography to perform secure transactions with remote machines.

2. Technical Issues

For DoD users working with any of the recent Microsoft Windows operating systems, CAC tools and documentation are readily available and are distributed by the Services, making client-side connectivity a fairly simple

process. Mac OS X is becoming more readily supported; in 2006 the U.S. Army added Thrusby's ADmitMac for CAC software to its official enterprise software list. Linux users are still left largely to fend for themselves, although this situation is slowly changing. At the Defense Information Systems Agency (DISA), for example, the Open Source Steering Group (OSSG) exists to tackle some of the basic issues regarding the convergence of Linux, CAC and PKI technologies. And as smart cards are used internationally for personal banking, government and corporate identity management, the OSS community has already devoted much effort to the cause of Linux-based smart card functionality.

The primary caveat to Linux interoperability is that CAC readers must be compliant with the PC/SC specification, which has become the de facto, cross-platform industry standard for smart card compatibility design. Compliance with PC/SC is also mandated by the Defense Manpower Data Center for all card readers used by the DoD. To operate in the Linux environment, the card reader must have Linux driver support. This can come in the form of a proprietary driver from the vendor, or compatibility with the free, open source USB CCID driver, courtesy of the Movement for the Use of Smart Cards in a Linux Environment (MUSCLE) project. From the http://musclecard.com website, “MUSCLE is a project to coordinate the development of smart cards and applications under Linux. The purpose is to develop a set of compliant drivers, API's, and a resource manager for various smart cards and readers for the GNU environment.” The project's PC/SC Lite middleware has also become ubiquitous in the Linux world as a means of allowing applications to communicate with smart card readers. One of the best resources for determining a card reader's Linux compatibility is the MUSCLE project's USB CCID site, at http://pcsclite.alioth.debian.org/ccid.html, which contains names and photographs of many devices.

According to the OpenSC project (http://www.opensc-project.org/opensc/wiki/OverView), one needs the following software tools to make use of a smart card: an application, a library, middleware and a driver. The

application might be a web browser like Firefox or an email client like Novell Evolution. As previously discussed, the middleware and drivers can be provided by the MUSCLE project's PC/SC and USB CCID tools, respectively. Finally, the library used in Linux is known as PKCS (Public Key Cryptographic Standards) #11, or the Cyrptoki Application Programming Interface (API), and is part of the Mozilla foundation's Network Security Services (NSS). The two free, open source PKCS#11 provider libraries are OpenSC and Coolkey. As a point of reference, the comparable Microsoft library is Crypto API. These libraries allow applications to interface with smart card tokens without having detailed knowledge about the card or reader hardware. Additionally, as NSS is Federal Information Processing Standard 140-2 validated, it meets requirements for unclassified cryptographic modules used by the U.S. Federal Government.

For the purposes of this study, RHEL 5.1 Desktop operating system was used in conjunction with its bundled Mozilla Firefox 1.5 web browser and Novell Evolution 2.x e-mail client. Further technical information on the procedures used to set up these clients for use with the CAC can be found in Appendix A of this document. In short, by copying the Mozilla Network Security Services (NSS) modules provided by OSSG to the appropriate directories on the client machine, all DoD Root Certificate Authority information is loaded to the client software, and the proper PCKS#11 libraries are installed as well. By completing this relatively simple task, users can access CAC-enabled websites with Firefox, and send and receive digitally signed and encrypted e-mail with Evolution. It should be noted that users will still need the public keys of e-mail recipients in order to send encrypted email. An enterprise-wide listing of all available recipients' keys is available at the DISA Global Directory Service (https://dod411.gds.disa.mil), where keys can be downloaded, then easily imported into any email client of choice. Finally, once the NSS modules are properly loaded into the system, OpenOffice.org can also take advantage of them to digitally sign documents (saved in Open Document format) using the CAC.

C. OTHER SECURITY BENEFITS AND CONCERNS

1. Vulnerability and Exploit Overview

Introducing a number of Linux-based workstations with various OSS applications into an otherwise homogeneous Windows network brings the added benefit of malware tolerance. Just as diversified crops are less susceptible to a strain of blight, employing a heterogeneous mix of operating systems and applications imparts a significantly greater level of virus, trojan and worm resistance to a corporate network. OSS systems have also historically shipped with default configurations that are inherently more secure than those of their Microsoft counterparts, providing less open ports and services for potential exploits, and giving users less system-wide privileges by default. Finally, developers of many popular OSS packages have proven adept at releasing vulnerability fixes before exploits are widely available.

Based on the evidence available at this point, however, one cannot objectively state that OSS is or is not inherently more or less secure than its closed source equivalents. The “many eyeballs” theory asserts that, since more people are examining the source code, flaws will be more readily discovered and fixed. The counter-argument says that attackers also have access to the same code, simplifying their task of finding faults and exploiting them first. Additionally, the “many eyeballs” of code reviewers may more likely be looking for functionality bugs than security holes.

In August of 2007, Jack Germain of the LinuxInsider technology website published a two-part story comparing the current state of open- and closed-source web browser security. His work focused on zero-day browser exploits: those attack vectors which take advantage of vulnerabilities that were previously unknown to system owners, defenders and the user community at large. These chinks in the software armor, instead of being exploited immediately, are sometimes bought and sold on the malware black market, and are capable of fetching tens and perhaps hundreds of thousands of dollars per exploit (Miller,

2007). According to Germain's research, “Vulnerability management solutions firm PatchLink sought a closer view of its customers' concerns over browser security issues in a recent survey. Responses from 250 customers revealed that the No. 1 security concern was zero-day vulnerabilities, Paul Zimski, director of product and market strategy at PatchLink, told LinuxInsider.”

Germain rather understandably did not reach a conclusion or proffer an opinion on which type of software (OSS or closed-source) was more secure. But due to the target-specific nature of zero-day exploits and the fact that they are so difficult to defend against, the only sure-fire countermeasure is to employ a heterogeneous mix of systems, ensuring that at least some network hosts are impervious to the attack. This kind of thinking should resonate with DoD network security managers as part of a “defense in depth” strategy, as the Department cannot afford to lose complete network functionality due to the sudden appearance of one zero-day exploit.

2. Transparency and Backdoors

While neither OSS nor closed-source software may ever be free of inadvertent coding-error-related vulnerabilities, only OSS offers the transparency necessary to instill confidence that the software has no intentional backdoors. Whether software vendors include backdoors for simplified system maintenance or for more nefarious purposes, they may eventually be discovered and exploited by interested third parties. A more benign example of this type of programming is known as the “easter egg.” Famous examples include the simple flight simulator and car racing games found in older versions of Microsoft Excel. These sub-programs, unrelated to the title application, can be accessed by performing a secret series of key presses and mouse clicks. The games are not harmful to the system (other than unnecessarily using up system storage space), but serve to illustrate how simple it can be to hide code within a closed source program.

Making this scenario more disconcerting is the rampant subcontracting and off-shoring witnessed in today's software development environment. While a

primary vendor may have the best intentions, subcontracted code modules, inadequately vetted, can introduce significant security risks. This should be of particular concern to any DoD agency; as the off-shoring trend is unlikely to be reversed, it must be factored into software security policies and strategies.

In certain conditions and to certain customers, Microsoft makes the source code for its Windows operating system and Office productivity suite available under the Shared Source Initiative. However laudable, this offering has several shortcomings that keep it from being truly open. First, only select users can obtain the source code for review. For example, certain governments may view the Windows code base under the Government Security Program (GSP). Code access is tightly restricted via smart card authentication. This limited release significantly reduces the “many eyeballs” type of security enjoyed by fully open source software. Additionally, the code is available for review, but may not be modified in any way (to meet specific needs or fix bugs, for example). Finally, reviewers may not compile the code into binary form, which would otherwise let them produce executable versions of the programs in question, identical to those delivered by the vendor. Without questioning the integrity of Microsoft's programmers, it is worth mentioning here that the entire development chain, including the source code of the compiler program, must be open to scrutiny to assure trust in the open source system.

In the strictest sense, though, even this is not enough. Unix programmer Ken Thompson brought this lesson to light in his 1983 Turing Award Lecture *Reflections on Trusting Trust.* In this famous treatise, he described a means of propagating a self-replicating trojan by slipping malicious code into the compiler program. Whenever the compiler was used to compile its own original source, the trojan was inserted into the new compiler binary. Thus, the source code of the compiler could pass muster under examination, even though it too would become a tainted binary once compiled. This type of attack might be defeated with the “Diverse Double-Compiling” (DDC) check, a method detailed by David A

Wheeler, but only if the reviewer has access to two independent compilers and their respective source code. As Wheeler puts it:

> . . . there's a catch: the DDC defense only works if you can get the source code for your software creation tools, including the operating system, compiler, and so on. That kind of information is typically only available for OSS/FS programs! Thus, even in the case of the dangerous "trusting trust" attack, OSS/FS has a security advantage (Wheeler, 2007).

3. Department of Homeland Security Code Scan

Although it is a minority opinion, detractors have contended that OSS is unfit for government and military service because we cannot know who is contributing to the source code (Wolfe, 2004). This argument would only hold water if OSS were instead "closed source with random public contribution" software. Fortunately, it is not. Contributions are not blindly added to mainline releases, but are scrutinized by application and kernel owners and any number of interested third parties (perhaps to include the utilizing agencies themselves) before they are published. After the new code has been publicly released (if it has even been released in binary form; much OSS is released in source-only form), the source code is still permanently available for examination by any curious individuals. Additionally, most prominent OSS projects manage code contributions with some variant of the Concurrent Version System (CVS). These systems track contributions as an inherent part of their functionality, making unnoticed malicious changes even less likely.

Claims of intentional coding malice aside, OSS is as susceptible to human error as any other software, even with its more open peer review. Whereas proprietary software vendors might pay to have mission-critical code scrutinized by third party firms and tools, OSS developers have had, in some cases, neither the resources nor the motivation to do the same. However, several prominent OSS packages have become a vital part of the Internet as we know it; calling them "mission-critical" to the daily business of the nation is not an exaggeration.

In 2006, the Department of Homeland Security recognized this fact and awarded a research grant to Stanford University, Symantec Corporation and Coverity, a company that specializes in software development. The aim of the 1.2 million dollar, three-year project was static analysis of the source code of 40 of the most prominent OSS packages using Coverity's Prevent software package. Results are provided to the OSS software maintainers so that bugs can be quickly corrected. As of the writing of this document, the http://scan.coverity.com project website claims that almost 8,000 bugs have been fixed since early 2006.

D. BOOTABLE "LIVE" OPERATING SYSTEMS

Several Linux distributions are released (or modified by third parties) so that the entire operating system can be run from the installation media, be it CD, DVD, USB memory, or other type of digital storage. No files are installed to the host computer's hard drive; the entire system is loaded to and run from Random Access Memory (RAM). Known as Live CD's, these distributions present an interesting option for easily-revised, quickly distributed, complete bundled operating environments. They may be a perfect solution for remote/mobile classified computing, where a user boots the CD and establishes an encrypted session with a remote classified server. When finished working, he or she then removes the CD and simply reboots the laptop, leaving no classified material on that machine. The client system need not even be fitted with a hard drive, to ensure that no trace of classified data exists when the system is powered-down.

As an aside, Live CD's present an excellent opportunity for both administrators and users to gain familiarity with OSS operating systems and applications without disturbing existing Windows installations. Users could even be given a Live CD of the enterprise's standard OSS desktop configuration to test and use at home in a risk-free manner.

E. APPLICATION AND CODE RE-USE

One intangible benefit of OSS is its potential for re-use throughout related communities of interest. Many government agencies perform similar functions,

only at different levels or across different geographic regions. Instead of spending millions of dollars to procure the same tools over and over again, these agencies might take advantage of OSS code that has already been developed or modified by another agency for the same purpose. The National Center for Open Source Policy and Research has, through a site called http://GovernmentForge.org, laid the groundwork for this kind of e-government OSS re-use in the civilian public sector. In the DoD, areas of potential overlap include personnel and finance systems, crew scheduling databases, maintenance record databases and more general-purpose computing tools such as web portals and office automation systems. These software needs are not unique to any one unit, command, or service, so why should these entities work independently to purchase or develop overlapping solutions? To sweeten the deal, most citizens may be pleasantly surprised to find that a number of their government's OSS development expenditures and contributions could be made available for use by the public at large. Such was the case with the National Security Agency's work on a Mandatory Access Control version of the Linux kernel, released to the public as Security Enhanced Linux (SE Linux). SE Linux code is now present in many popular Linux distributions, to include RHEL.

Shared libraries have long been used in the programming world to help reduce duplication of effort. The OSS community has taken this concept to a macro-level to develop means of dealing with unique, nuanced requirements that intrude on an otherwise shared purpose. Instead of developing custom-built solutions for each requirement, a modular framework is first established around a common core. Plug-ins or modules are then written to address specific needs. This paradigm can be witnessed in action in the Apache web server and the Linux Pluggable Authentication Module (PAM) frameworks. Especially where this modular framework is concerned, the potential benefits of such code re-use are even more apparent.

Standardization and reusable components have historically been key to moving hitherto artisan practices into the streamlined, efficient world of

industrialization. In software development, rapid standardization has been realized by means of open communication via the Internet. OSS and open standards themselves have seen growth and innovation that would likely not have been possible in numerous smaller, sequestered work environments. And OSS has been able to progress at its current pace largely due to the availability of quality, reusable, interchangeable components. Such collaborative potential is a significant factor that must be tied back into TCO for use in a “best value” procurement decision.

IV. POTENTIAL OPEN SOURCE SOFTWARE CHALLENGES

A. REGULATIONS GOVERNING SOFTWARE USE

1. Information Assurance

Although DoD computer systems are not limited to running one particular operating system from any one vendor, each of the Services must observe regulations which restrict the information systems software and hardware that may be attached to the DoD's official networks. As described briefly in Chapter III of this document, DoD-level doctrine exists to establish technical definitions and Department-wide standards for the use of networked operating systems, which fall under the category of Information Assurance (IA) or IA-enabled Information Technology (IT) products. According to DoD Directive 8500.01 (p. 18, section E2.1.21), IA-enabled IT is a "product or technology whose primary role is not security, but which provides security services as an associated feature of its intended operating capabilities. Examples include such products as security-enabled web browsers, screening routers, trusted operating systems, and security-enabled messaging systems".

DoD Directive 8500.01 then mandates that all IA-enabled IT products must "comply with the evaluation and validation requirements of National Security Telecommunications and Information Systems Security Policy Number 11" (NSTISSP-11). That document, now maintained by the Committee on National Security Systems (CNSS), specifies that "the acquisition of COTS IA and IA-enabled IT products (to be used on systems entering, processing, storing, displaying, or transmitting national security information) . . . shall be limited only to those which have been evaluated and validated in accordance with the criteria, schemes, or programs specified . . ." Those programs are listed below:

- The International Common Criteria for Information Security Technology Evaluation Mutual Recognition Arrangement (CCRA). The CCRA is a

mechanism by which international member government agencies can take advantage of evaluations that were performed in different countries. While useful, the CCRA limits internationally recognized evaluations to EAL 4.

- The National Security Agency (NSA) / National Institute of Standards and Technology (NIST) National Information Assurance Partnership (NIAP) Evaluation and Validation Program. The NIAP Evaluation and Validation Program is the Common Criteria process in the United States.
- The NIST Federal Information Processing Standard (FIPS) validation program. The FIPS validation program is required for any technology that provides cryptography for United States government information systems.

Where cryptography is concerned, it is important to note that an entire system does not need FIPS certification, only the cryptographic modules it uses. In the case of desktop software, this usually boils down to the components that handle digital signature and encryption. For many Linux systems, Secure Sockets Layer (SSL) and Transport Layer Security (TLS), perhaps the most commonly-used cryptography standards on the Internet, are provided by the OpenSSL libraries. The Mozilla Firefox web browser, Thunderbird e-mail client, OpenOffice.org, and other applications use the Network Security Services (NSS) libraries, which provide SSL, TLS and various Public Key Cryptography Standards (PKCS). PKCS implementations bring digital signature, encryption and smart card capabilities to any program written to take advantage of the PKCS libraries. Both NSS and OpenSSL are FIPS 140-2 certified.

All of these rigid regulations governing software use in the DoD can significantly complicate the process of integrating any typical proprietary software into a DoD network, and this applies to OSS as well. While the rules help to provide a secure, standardized baseline for information technology products on the Global Information Grid, they also serve as barriers to entry for smaller software projects. Common Criteria and FIPS validation can cost from tens to hundreds of thousands of dollars and take years to achieve (GAO, 2006, p. 19).

The time factor can keep otherwise mature products out of federal institutions, and is also directly at odds with the rapid development models used by most software development, and especially OSS projects. “This disconnect between industry and NIAP has resulted in an awkward evaluation process that ensures that security products are well into their life cycles, if not obsolete, by the time they can be evaluated, vendors say” (Jackson, 2007).

The financial factor is an obvious barrier to entry for any community-maintained software suite. In the OSS world, this has sometimes been overcome by either direct sponsorship from a utilizing agency, or third party support from an interested vendor. In the case of OpenSSL, the U.S. Defense Medical Logistics Standard Support (DMLSS) spearheaded the initiative to obtain FIPS 140-2 validation. According to comments from Debora Bonner, Director of Operations at DMLSS (http://www.linuxdevices.com/news/NS4742716157.html),

> The DMLSS program is heavily dependent on OpenSSL based cryptography, so this validation will save us hundreds of thousands of dollars," Bonner added. "Multiple commercial and government entities, including Medical Health Systems (MHS), have been counting on this validation to avoid massive software licensing expenditures. The three year validation process was an ordeal, but our persistence finally paid off.

As a result of this one effort by DMLSS and others, numerous government agencies are now able to legally employ free OpenSSL cryptography in federal programs.

2. Licensing

The Open Source Institute lists several dozen “open” software licenses (http://www.opensource.org/licenses/category). Of these, nine are categorized as popular or widely used. Among these nine, the most prevalent in the OSS community is the GNU General Public License (GPL). The GPL has been termed a “viral” software license, in that derivative works must also be licensed under the GPL, which serves to maintain the open source process. Additionally, there has been some concern that GPL-licensed products might not be suitable

for government use and modification, since any publicly released, derived programs must have their source code made available as well. This may be a valid concern in some cases. In the case of generic, non-sensitive programs, releasing source code fixes and/or modifications would serve to benefit the original program and perhaps the nation at large. However, for sensitive projects, the resulting binary programs should not be made publicly available anyway, and therefore the modified source would not be an issue of concern, either. This all hinges on whether or not the GPL-derived, modified code remains within the organization. Per the http://www.GNU.org Frequently Asked Questions page,

> The GPL does not require you to release your modified version, or any part of it. You are free to make modifications and use them privately, without ever releasing them. This applies to organizations (including companies), too; an organization can make a modified version and use it internally without ever releasing it outside the organization.

Finally, it should be noted that OSS licensed under the GPL (or similar licensing for software meeting the Open Source Initiative's definition) is not public domain freeware/shareware, specifically because of its unique licensing. However, DoD Directive 8500.01 (p. 6, Section 4.19) makes an interesting (if unfortunate) link between public domain software (freeware) and products with limited or no warranty:

> Public domain software products, and other software products with limited or no warranty, such as those commonly known as freeware or shareware, shall only be used in DoD information systems to meet compelling operational requirements. Such products shall be thoroughly assessed for risk and accepted for use by the responsible DAA [Designated Approving Authority].

This excerpt from DoDD 8500.01, while well intentioned, places perhaps undue emphasis on warranty, and may dissuade the use of OSS in some cases where it is the best option. Many pieces of OSS are offered with limited or no warranty. This includes software distributed under the GNU General Public

License, and most other popular OSS licenses. By comparison, many popular pieces of proprietary software are offered with either no warranty (and, in fact, vendor indemnification clauses), or very little warranty. For example, the Microsoft XP End User License Agreement (EULA) offers a limited, 90-day warranty against defects, but the warranty does not extend to service packs or other hot fixes applied after the initial 90-day period (http://www.microsoft.com/windowsxp/sp2/proeula.mspx). Additionally, the EULA specifies that users are not entitled to any damages whatsoever if the software in any way fails to perform as expected. Therefore, Windows XP is delivered with a warranty, but most of it seems to serve the vendor more than the user. Again, this only serves to highlight the perhaps negative (and questionable) link between free software and warranty in DoDD 8500.01.

3. Further Intellectual Property Issues

In 2003, the SCO Group sued IBM and threatened Linux users around the world, stating that they (SCO) owned the Unix copyright, and that Linux was infringing on this copyright by illegally using Unix code. To date, these allegations have not been publicly proven. However, several large ISV's, to include Red Hat, HP and Novell, have offered their Linux customers indemnification against any potential copyright or patent infringement lawsuits. The Red Hat Open Assurance program, for example, also includes replacing/modifying any potentially infringing code, or obtaining the rights for a customer to continue using the code legally, should such an issue arise. It now appears unlikely that users of Linux code will face such copyright infringement concerns (the SCO Group filed for Chapter 11 bankruptcy protection in late 2007), but the stated protections offered by large Linux systems vendors may remain valuable to potential government customers.

B. COMMUNITY-SPECIFIC SOFTWARE NEEDS

1. Open Source Software Modification

Numerous OSS applications have reached levels of capability and maturity that have already made them dominant tools for their respective jobs. Examples include the Apache web server, MySQL database, Sendmail e-mail server, BIND Domain Name System and Linux operating system (at least in a server capacity). Still, OSS chosen for use by DoD agencies may require modifications beyond what the standard commercial packages offer. And while it is true that the availability of source code makes OSS inherently more adaptable than closed-source software, there are costs involved in taking advantage of this benefit beyond the price of hiring programmers to modify the code. Most notably, interested entities must deal with the dilemma of forking, which occurs when a modified piece of code is not reintroduced into the "mainline" software distribution. Instead, the mainline code authors continue along their original path, making modifications and releases independent (perhaps even unaware) of the forked product. As a result, the agency which has forked the code in its own direction may be incapable of taking advantage of subsequent mainline advances such as bug-fixes, security patches and other feature updates. An eventual solution to these concerns may be a series of Government Off-the-Shelf (GOTS) OSS products with COTS roots.

An example, albeit a vastly simplified one, can be found in Scientific Linux. This Linux distribution was created by the Fermi National Accelerator Laboratory and the European Organization for Nuclear Research (CERN). The distribution's creators have taken the publicly available source code of Red Hat Enterprise Linux, compiled it, and made several additions and changes to suit their own needs. However, they state very clearly that their distribution will always be binary-compatible with the corresponding Red Hat release. In this way, any software compiled for Red Hat systems can also be seamlessly run on Scientific Linux. It is a simple logical step to apply this concept to the DoD's needs. Each

branch of service (and/or major command within that service) could have its own distribution, populated with special-purpose programs, public key certificates, and even document libraries that meet that community's needs. Updates would be centrally controlled, undergoing testing for security vulnerabilities and other flaws, but essentially keeping in line with open source community releases.

2. Porting and Application Compatibility

When an acquisition authority considers supplanting existing, familiar systems with new OSS alternatives, he or she must determine and weigh all the support-related costs of migrating existing capabilities to the new systems. In some cases, this may be done with little turmoil if users can accomplish the same tasks natively in the new environment. However, where legacy applications are in heavy use and have no apparent equivalent on the new OSS platform, they may have to be ported to the new architecture to enable continued productivity. In short, porting typically entails significant software development; rewriting the code that was intended to run on one platform so that it can be natively executed on another platform. This can have a considerable negative impact on both cost and schedule and may even be detrimental to performance depending on the quality of the code re-engineering.

To tackle the challenge of migrating Windows applications to the Linux-based desktop, one alternative to porting is use of the Wine (Wine Is Not an Emulator) environment. As its name implies (and its developers are keen to point out), Wine is, in fact, a Windows application compatibility layer, not an emulator. It translates Windows instructions into those understood by Linux, allowing users to install and use many (though not all) Windows applications in a Linux environment. The application database (http://appdb.winehq.org/) contains a fairly comprehensive list of Windows programs that have been tested by the user community, along with a score that corresponds to each program's level of functionality in Wine. Wine is one example of how the DoD's efforts to contribute to open source projects might serve an incredibly large audience both inside and outside the U.S. government. For example, the ICS Viewer, which is discussed

in Chapter II, Section A5, is somewhat functional under Wine. Many official U.S. Air Force forms can be opened and viewed with ICS Viewer under Wine on a Linux system (Figure 1), but text entry is problematic at best. DoD code contributions to Wine might resolve not only this, but various other problems experienced by related Windows programs in the Wine environment.

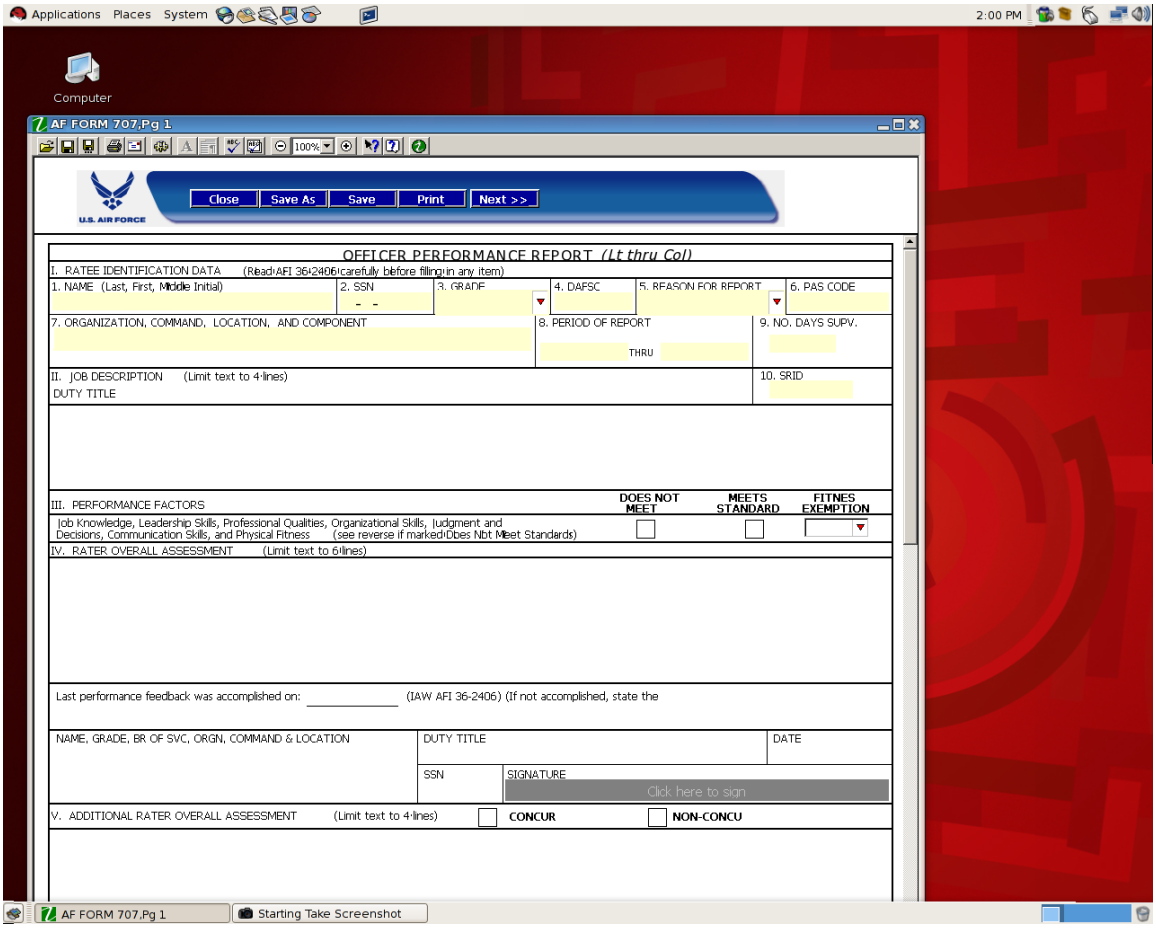

Figure 1. ICS Viewer in Wine on Red Hat Linux

3. Virtualization and Centralized Computing

This thesis has focused largely on integrating OSS-based systems into Windows environments, rather than suggesting widespread system replacement. However, increasing the number of primarily OSS-based desktops in the enterprise reduces the number of licenses required for Microsoft Windows and any proprietary software that runs on it, such as Microsoft Office, Adobe Acrobat, etc. The goal of proprietary license reduction need not conflict entirely with the needs of specific users who take full advantage of features that may only be available on Windows-based systems. Several technologies exist to allow efficient use of a limited number of proprietary software licenses. These include virtualization, remote/centralized computing, and application streaming.

"Virtualization" is perhaps the loudest buzzword in desktop computing technology in 2008. It is a technique whereby a guest operating system can be run inside a virtual environment, controlled by a "hypervisor" or "virtual machine monitor." Until quite recently, the hypervisor on desktop systems had to be run inside a complete host operating system (Figure 2), a technique which severely limited the speed and scalability of guest systems. However, currently available workstation and server processors contain the ability to allow hardware-assisted virtualization, whereby system resources are shared to the hypervisor at a hardware level, sometimes with no full host operating system running underneath the guests. This greatly increases speed and efficiency, and allows for better sharing of the hardware resources between the guest operating systems.

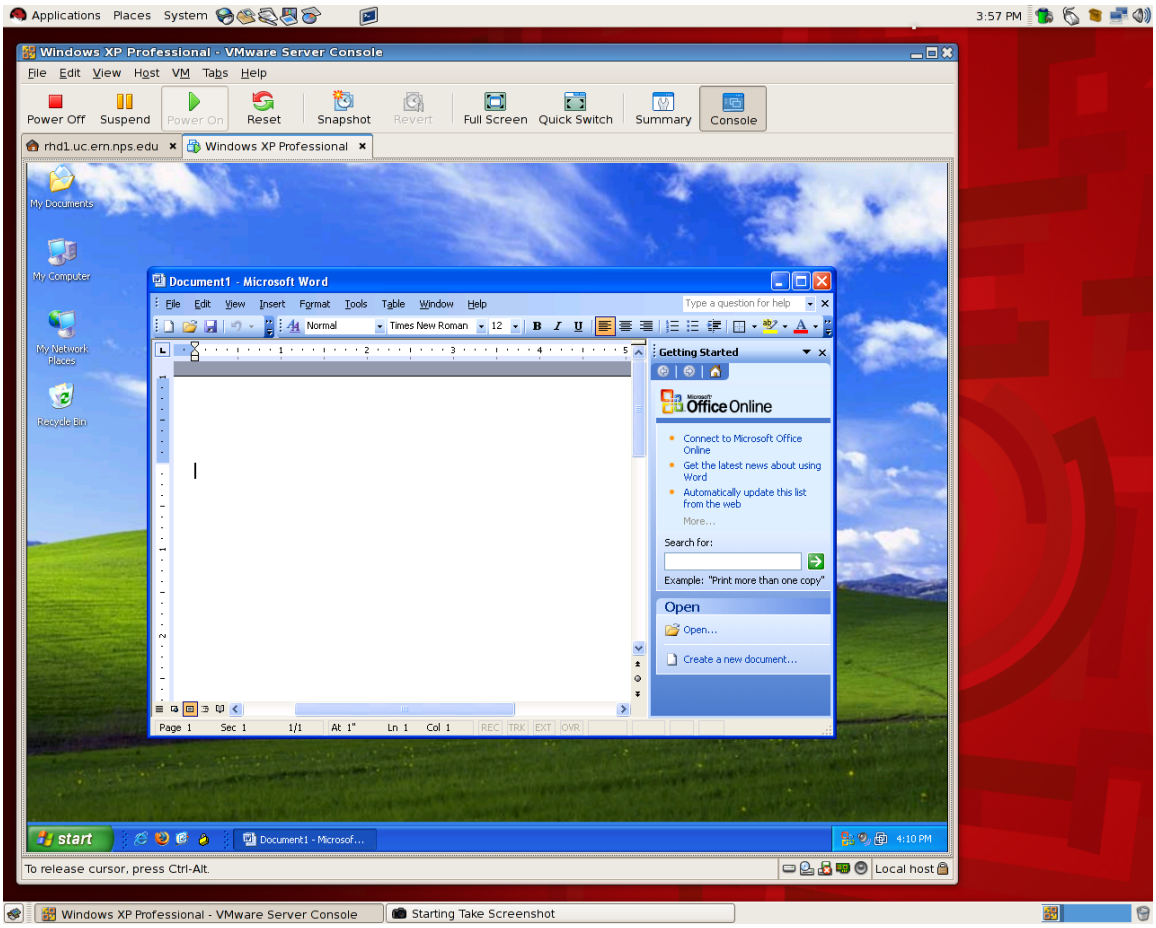

Figure 2. Windows in VMware on Red Hat Linux

Traditional centralized desktop computing, known as the “terminal server” system, provides on-demand access to a complete operating environment, entirely hosted on a server and delivered live over the network. This remote window to the operating environment places great demands on the network, and is known for its user-unfriendliness where rich media is involved. Screen updates can be painfully slow, such that working with simple presentations is difficult, and multimedia playback (let alone editing) is not advisable. For many core business applications, though, terminal server setups offer a cost-effective way for a subset of users to access Microsoft Windows profiles from any device that has a remote desktop client (Figure 3).

Figure 3. Windows Remote Desktop from Linux

Hybrid solutions now exist to address the shortcomings of remote computing, incorporating a melding of both remote desktop and virtualization technologies. The recently released XenDesktop product from Citrix is one such example (Figure 4).

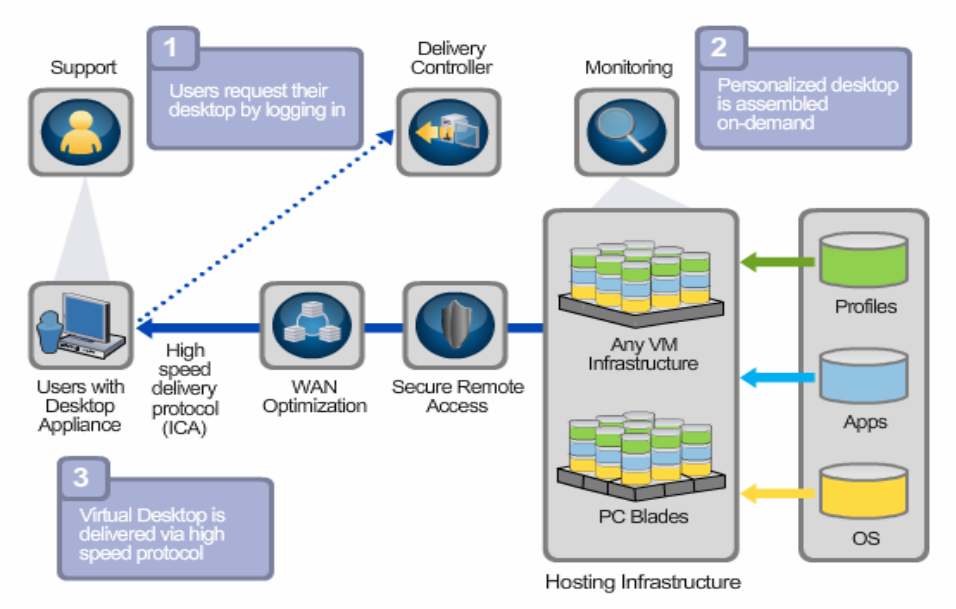

Figure 4. Citrix XenDesktop Virtualization

Source: http://www.citrix.com

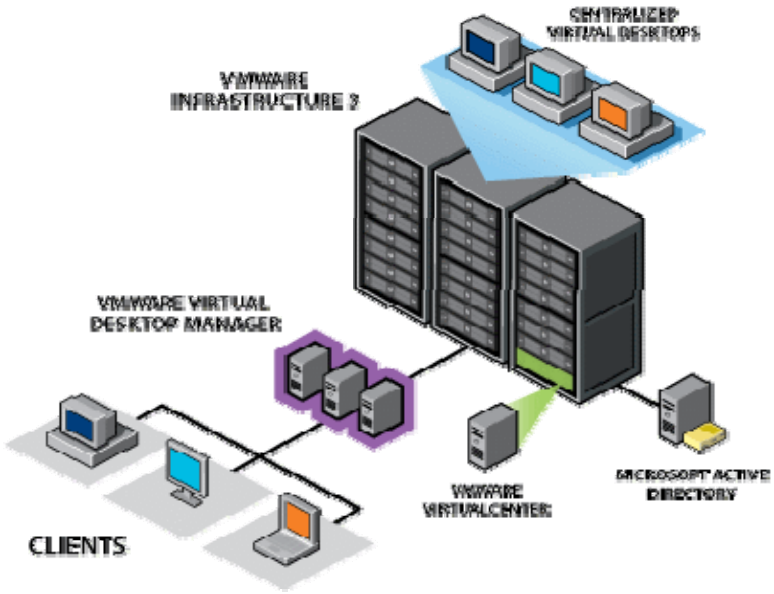

Figure 5. Vmware Virtual Desktop Infrastructure

Source: http://www.vmware.com

C. HETEROGENEOUS MIX AND SUPPORT CHALLENGES

1. End Users

If users are expected to migrate to new applications in a foreign OSS environment, they may require retraining, bringing a significant new cost into the equation. Authors of OSS desktop software have been increasingly keen to avoid this problem. Their recent work has demonstrated an understanding that, in order to spur adoption, they must design their programs to operate (at least on the surface) in a manner similar to their Microsoft Windows counterparts. In fact, the general "look and feel" of the Windows Graphical User Interface (GUI) has been all but replicated in a majority of Linux distributions such as RHEL, Novell's SUSE Linux, Canonical's Ubuntu Linux and others. Intermediate-level Windows users can very likely accomplish general productivity tasks in these OSS operating systems with little or no familiarization training. "Whilst the first versions of Linux were fairly difficult to use for non-technicians, the product is widely considered to have matured at the end of the 1990s and now there is no significant difference in terms of ease of use between Windows and most commercial Linux operating systems" (European Commission, 2004, p. 127). The same holds true for OpenOffice.org, the primary OSS competitor to the Microsoft Office suite.

2. Systems Administrators

Initial purchase costs are rather simple to determine and incorporate into an overall budget, but they only account for a fraction of the overall system life cycle costs. Approximately 60 to 80 percent of a program's software costs fall in Post Deployment Software Support (Petross, MN3331 lecture, Winter 2008). While having a heterogeneous mix of systems may offer security benefits, it almost certainly comes at a price.

All claims of simplicity and compatibility of OSS described throughout this document are highly reliant upon an end user operating environment that has been expertly crafted and maintained by a truly skilled group of systems

integrators and administrators. While this is also true of most proprietary software networks, there is indeed a need for a different skill set, especially in the case of Linux systems administration. Again, the common themes of strong centralized standardization and control are paramount. Fortunately, DoD networks are evolving in this direction more every year, so adding OSS into this structure might not be so daunting a challenge as it would have been a decade ago.

A separate systems management framework also can present additional costs both in physical systems and administrator training. This issue cannot be swept aside or taken lightly, but it is also broad enough to warrant its own separate review outside of this document. Fortunately, OSS systems management options have matured beyond their simple command line roots into much more user-friendly (and GUI-based) tools. The command line interface (CLI) is still an option if administrators prefer it, but programs like Webmin make user management and other systems administration a point-and-click affair. Similarly, the Red Hat Network (RHN) makes it rather trivial to manage a large number of systems via a web interface, providing software installation and removal, hardware inventory, and other tools (Figure 6).

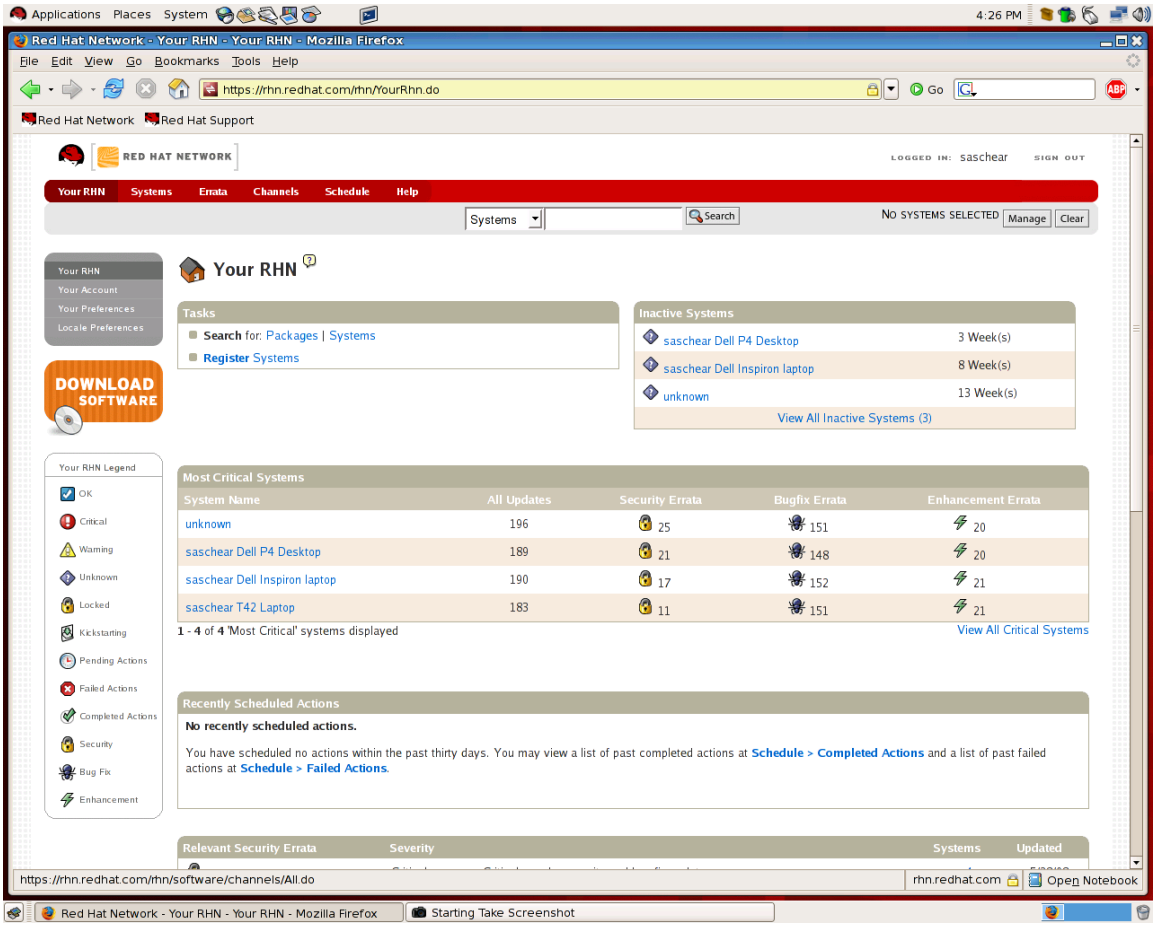

Figure 6. The Red Hat Network

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS

A. RECOMMENDATIONS TO DEPARTMENT OF DEFENSE INFORMATION TECHNOLOGY MANAGERS

1. Official Guidance Overview

Several years ago, a MITRE report stated that "at present, FOSS is neither approved nor disapproved in most parts of the DoD. This limbo status makes program, project, and developer decisions regarding FOSS difficult" (MITRE, 2003, p. 22). Fortunately, in the intervening years, several national, DoD and service-level policies have been released which clarify the status of OSS, put it on a level playing field with proprietary, closed-source software and provide relevant guidance for acquisitions professionals.

At the national level, The Office of Management and Budget (OMB) reminds senior procurement executives that OSS licenses differ from proprietary software licenses and "may affect the use, the security, and the total cost of ownership of the software and must be considered when an agency is planning a software acquisition" (U.S. OMB, 2004). In the DoD, Chief Information Officer John Stenbit released a memo to the services in 2003 stating that "DoD Components acquiring, using or developing OSS must ensure that the OSS complies with the same DoD policies that govern Commercial off the Shelf (COTS) and Government off the Shelf (GOTS) software" (U.S. DoD CIO, 2003). Finally, the U.S. Navy's Chief Information Officer wrote "A key piece in supporting the DoD goal is the ability to utilize OSS as part of the Department of the Navy's (DoN's) Information Technology (IT) portfolio" (U.S. DoN, 2007). These policy letters serve to reinforce the status of OSS as a viable alternative in DoD software development and acquisitions.

2. Open Architecture in Government Contracting

For the software-savvy acquisitions professional, the thought of employing OSS in any given project most likely summons conflicting feelings of elation and apprehension. Elation comes from the prospect of low or non-existent initial and recurring licensing fees for the software. But apprehension creeps in at the thought of all the unknowns. Will the existing workforce be able to easily and effectively use and/or maintain the system? If not, how much training will be required? What kinds of impacts will the OSS have on interoperability, both within the organization and with external systems and users? The answers to these questions and many others, when combined with raw dollar purchase prices, will help to determine a system's Total Cost of Ownership (TCO). Proprietary software vendors often note the aforementioned questions as drivers of potentially higher TCO when considering OSS. Indeed, these factors must be weighed against benefits to bring the acquisitions authority closer to the desired outcome of "best value" in government contracting. In the end, the decision about whether or not to employ OSS for a given task will depend on whether or not it is the best tool for the job, when all factors (including those addressed in this document) are taken into consideration.

OSS has been (perhaps unwittingly) given a second-tier status by some acquisitions authorities, although it often comes closer to the standards that are set for weapon system procurements. When the DoD prepares a multi-billion-dollar contract for aircraft or vehicle construction, it mandates that the vendor must supply technical drawings and schematics along with the delivered hardware. The same applies to construction or upgrades of military facilities. This is done in order to ensure that internal, organic maintenance units, as well as contracted third parties, can perform any required service on the vehicle or structure. Without such supporting documentation, DoD agencies would be indefinitely locked into one vendor for weapon system support, a concept that is shunned in the acquisitions community. Yet somehow, this standard does not carry over to the desktop computer software world. Vendors typically deliver

proprietary code that cannot be extended, updated or otherwise modified by a third party, precisely due to lack of source code and adequate documentation.

In September of 2007, the Defense Federal Acquisition Regulation Supplement (DFARS) was amended, instructing contracting officers to more carefully consider DoD long-term needs for technical data rights for weapon systems. This was mandated specifically to allow a strategy that includes "the development of maintenance capabilities within DoD; or competition for contracts for sustainment of the system or subsystems" (DFARS 207.106). Additionally, the supplement states:

> Although the law does not address requirements for computer software, it is long-standing DoD policy to apply the same or similar requirements to both technical data and computer software, since many issues are common to both. Therefore, this interim DFARS rule applies to both technical data and computer software.

The "technical data rights," translated in preceding paragraph to also cover computer software rights, could reasonably be interpreted to include program source code and access to any API documentation that the DoD might require to maintain the software and/or allow for future sustainment contracts through competing corporations.

The U.S. Navy's Program Executive Office for Integrated Warfare Systems has taken a leadership role in promoting open architectures in DoD contracting, and its Naval Open Architecture (NOA) Guidebook for Program Managers also addresses the need for government rights to technical data:

> NOA is the confluence of business and technical practices yielding modular, interoperable systems that adhere to open standards with published interfaces. This approach significantly increases opportunities for innovation and competition, enables re-use of components, facilitates rapid technology insertion, and reduces maintenance constraints. NOA delivers increased warfighting capabilities in a shorter time at reduced cost. The U.S. Government's (hereinafter "Government") ability to acquire at least Government Purpose Rights (GPR) to data and intellectual property and to minimize proprietary elements to the lowest component level is critical to this effort.

While it is true that open architecture does not directly equate to open source, the two tend to go hand-in-hand. Most OSS packages are written around open communications standards and file formats. Additionally, when open standards are used throughout an enterprise, it simplifies the further integration of OSS. Therefore, IT managers are advised to employ open standard technologies wherever possible.

Web-based applications that conform to World Wide Web Consortium (W3C) standards offer cross-platform capability that can be centrally maintained and updated, with no client-side installation or maintenance effort. Other technologies that treat software as a service, and can do so with little regard for client platform, offer a similarly future-proof and centrally maintainable solution. Such solutions almost make the desktop operating environment irrelevant, so long as it conforms to open standards and provides a basic set of what are currently considered commodity operating functions.

B. RECOMMENDATIONS FOR FUTURE RESEARCH

This thesis includes a section on Common Access Card (CAC) technology and an appendix devoted to testing capabilities of Common Access Cards in OSS environments. However, due to the lack of a suitable testing environment, there is no survey of the ability of OSS-based clients to log into Windows/Active Directory domains using CACs. Several vendors, including Red Hat and Novell/SUSE advertise the ability to integrate Linux-based clients rather well into these environments. And as previously discussed, two-factor authentication (usually in the form of CAC login) is mandatory throughout the DoD for access to the NIPRNet. It would be of great value to continue this research by determining to what extent OSS-based client systems can take advantage of Windows domain logins (and associated controls, such as Group Policy Objects), particularly when using CAC authentication.

The full-scale integration of OSS systems goes beyond domain membership. Terry Bollinger noted in his 2003 MITRE report (p. 24), OSS "seems to work best when people come to it, and not vice-versa." This approach

may be acceptable for smaller-scale integration of specific tools in niche applications. However, it cannot be the case for large-scale, professional deployments of OSS-based desktop systems. Standardization is key to healthy, economically viable enterprise management. Following in the path of current desktop systems standard configuration efforts, OSS must be deployed in a similar manner. Exactly how this should be accomplished, to what extent, and by whom is a topic worthy of further study.

This thesis began with an observation that the DoD spends millions of dollars each year on typical desktop computer software. Alternatively, Open Source Software is often referred to as “Free Open Source Software” throughout the industry both because of the freedoms it affords users (modification, re-distribution, etc.) and the fact that it is usually available free of charge. Paradoxically, as discussed throughout this document, the DoD is often not able to take advantage of all such OSS, usually due to information assurance regulations. In the case of operating systems, vendors expend significant time, effort and money to bring OSS packages into line with DoD requirements. DoD agencies are then limited to those offerings. But must this be the case? On one hand, DoD agencies might be thankful (despite the financial cost for support licenses) that these vendors are both interested in tackling the challenge of software certification, and are also willing and able to provide follow-on support. Alternatively, the DoD might consider a series of internally certified and supported OSS systems, taking some of what is currently available off-the-shelf and using it as a starting point for DoD-specific variants. This might still be done in cooperation with OSS vendors, but at significant cost savings over typical, per-seat support licenses.

C. CLOSING THOUGHTS

OSS has received increased attention within DoD technology and acquisition circles in recent years, and with good reason. OSS offers solutions that can satisfy a majority of official business needs with (at least up-front) significant financial savings. Desktop OSS is already in use on niche classified

systems, and is generally accepted as a secure, transparent and trusted alternative for personal, corporate and government use worldwide. It can interface with the majority of Windows-based server and client systems widely used on DoD networks. It provides portable, modular code that lends itself to customization, and guarantees that using agencies won't be locked into any one vendor or platform. Many systems administrators and IT managers on the front lines of DoD networks understand these advantages, are familiar with numerous OSS products, and are eager to reap the benefits of those tools in an official, sanctioned capacity.

Chapter III of this document addressed the functionality delivered by the U.S. Air Force's Standard Desktop Configuration, and how currently available OSS tools might match those capabilities. In summary, OSS offerings provided what might be called the 90% solution; they would likely be adequate for a considerable number of users performing typical office automation and communication tasks. Unfortunately, the SDC does not represent the entirety of applications used by U.S. Air Force units. Almost every specialized Community of Interest (COI) utilizes some specific set of tools that are installed on top of the SDC. Some of the centralized computing tools described in Chapter IV are currently used to satisfy the needs of these COI's, namely remote desktop computing via terminal servers and Citrix-based application delivery. These solutions integrate painlessly into OSS desktop environments. The remaining OS-specific, locally installed applications may be problematic enough to keep OSS out of those offices for the time being. Then again, in this era of stretched military budgets, tools that provide the 90% solution at a fraction of the cost may prove sufficiently attractive to military leadership to warrant replacement of the trouble-making niche applications.

OSS offers the DoD (and other branches of government) many attractive alternatives to proprietary desktop computer software, but the potential comes with a significant number of caveats. OSS solutions are not simple drop-in replacements for current proprietary desktop systems. As this document has

described, they can capably fulfill most typical office automation needs with a high level of cross-platform compatibility and a relatively low level of user retraining. But the details of implementation must be carefully addressed one-by-one in a methodical systems development life cycle approach. Each of the Services has the resources to develop OSS test platforms with very low initial acquisition costs. These resources must be tapped and fully utilized to drive standardized, top-down deployments for successful, widespread OSS integration. If acquisition authorities decide to venture down the OSS path, the journey must be undertaken with a very high level of preparation and commitment, but the potential pay-off is exciting, to say the least.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Altman, S. W. (2003, Apr 7). Evaluating Potential Alternatives to Total Dependence on Microsoft for Desktop Operating Systems and Applications. Retrieved 2/27/2008 from http://stinet.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA415978.

Asiri, S. (2003, Mar). Open Source Software. *SIGCAS Comput. Soc.* 33, 1, Retrieved 3/1/2008 from. DOI= http://doi.acm.org/10.1145/966498.966501.

Boutin, P. (2006, Jul 3). *Where's My Google PC?* Retrieved 3/3/2008 from http://www.slate.com/id/2144896/.

Business Wire (2007, Jul 16). *Red Hat Enterprise Linux 5 Independently Certified at Common Criteria EAL4+ Under NIAP Scheme by atsec and HP.* Retrieved 2/4/2008 from http://www.businesswire.com/portal/site/google/index.jsp?ndmViewId=news_view&newsId=20070716005868&newsLang=en.

Center for Strategic and International Studies (CSIS) (2007, Aug 20). *Global Policies on Open Source Software.* Retrieved 2/27/2008 from http://www.csis.org/component/option,com_csis_pubs/task,view/id,4009/type,1/.

Committee on National Security Systems (CNSS) (2003, Jul). *National Information Assurance Acquisition Policy*. (National Security Telecommunications and Information Systems Security Policy (NSTISSP) No. 11 Revised Fact Sheet). Retrieved 2/27/2008 from http://www.cnss.gov/Assets/pdf/nstissp_11_fs.pdf.

Corbet, J. (2007, Feb 20). *Who Wrote 2.6.20?* Retrieved 1/20/08 from http://lwn.net/Articles/222773/.

Corbet, J., Kroah-Hartman, G., McPherson, A. (2008, Apr). *Linux Kernel Development (April 2008).* Retrieved 4/17/2008 from https://www.linux-foundation.org/publications/linuxkerneldevelopment.php.

Dowling, T. (2000). Software COTS Components – Problems, And Solutions? Retrieved 2/12/2008, from http://handle.dtic.mil/100.2/ADA394911.

European Commission (2004, Mar 24). Commission Decision of 24.03.2004 relating to a proceeding under Article 82 of the EC Treaty (Case COMP/C-3/37.792 Microsoft). Retrieved 3/4/2008 from http://www.microsoft.com/presspass/download/legal/europeancommission/03-24-06EUDecision.pdf.

Evers, J. (2003, Jan 15). *Microsoft Grants Governments Access to Windows.* Retrieved 4/5/2008 from http://www.pcworld.com/article/id,108804-page,1/article.html.

Germain, J. (2007, Aug 14). *Zero-Day Browser Exploits, Part 1 – Is Open Source Safer than IE?* Retrieved 4/5/2008 from http://www.linuxinsider.com/story/58796.html.

Humphrey, W.S. (1989). *Managing the Software Process.* Boston: Addison-Wesley.

IBM (2007, Oct 2). *IBM Brings Web 2.0 Usability Features to Web-Based Electronic Forms Processing*. Retrieved 2/4/2008 from http://www-03.ibm.com/industries/consumerproducts/doc/content/news/pressrelease/3164239123.html.

Ingo, H. (2004). *Open Life – The Philosophy of Open Source.* [Electronic Version] Retrieved 4/2/2008 from: http://www.openlife.cc.

Jackson, W. (2007). *Under Attack – Common Criteria Has Loads of Critics, but Is It Getting a Bum Rap?* Retrieved 2/4/2008 from http://www.gcn.com/print/26_21/44857-1.html.

Klein, A. (2008, Jan 24). The Complex Crux of Wireless Warfare. *The Washington Post.* [Electronic Version] Retrieved 3/4/2008 from http://www.washingtonpost.com/wp-dyn/content/article/2008/01/23/AR2008012303695_2.html?sid=ST2008012303998.

McMillan, R. (2007, Jun 18). *Red Hat Linux Gets Top Government Security Rating.* Retrieved 2/12/2008 from http://www.computerworld.com.au/index.php/id;306842912;fp;4194304;fpid;1.

Miller, C. (2007, May 6). *The Legitimate Vulnerability Market – Inside the Secretive World of 0-Day Exploit Sales.* Retrieved 4/5/2008 from http://weis2007.econinfosec.org/papers/29.pdf.

MITRE Corporation (2001, Oct). *A Business Case Study of Open Source Software.* Retrieved 2/4/2008 from http://www.mitre.org/work/tech_papers/tech_papers_01/kenwood_software/index.html.

MITRE Corporation (2003). *Use of Free and Open Source Software (FOSS) in the U.S. Department of Defense.* Retrieved 1/18/2008 from http://terrybollinger.com/dodfoss/dodfoss_pdf.pdf.

National Information Assurance Partnership (NIAP). *NIAP Approved Common Criteria Testing Laboratories.* Retrieved 2/4/2008 from http://www.niap-ccevs.org/cc-scheme/cctls/.

Olavsrud, T. (2003, Jun 3). *Defense Department Issues Open Source Policy.* Retrieved 2/12/2008 from http://www.internetnews.com/dev-news/article.php/2216311.

Petross, D. (2008, Feb). *Principles of Acquisition and Program Management.* Presented at a MN3331 lecture at the Naval Postgraduate School.

Seiferth, C. J. *Open Source and These United States.* Retrieved 2/27/2008 from http://skyscraper.fortunecity.com/mondo/841/documents/99-184.html.

Thompson, K. (1984, Aug). Reflections on trusting trust. *Communications of the ACM* 27, 8, 761-763. DOI= http://doi.acm.org/10.1145/358198.358210.

U.S. Defense Information Systems Agency (2006, Mar 28). *Unix Security Technical Implementation Guide, Version 5, Release 1.*

U.S. Department of the Air Force (2002). *AF-CIO Policy Memorandum 02-14; Acquisition of Information Assurance (IA) and IA-Enabled Information Technology (IT) Products.* (U.S. Air Force Chief Information Officer Memorandum). Retrieved 1/20/2008 from http://www.niap-ccevs.org/cc-scheme/pm02_14_aq_ia_it_products.pdf.

U.S. Department of Defense, Chief Information Officer (DoD CIO) Memorandum (2003, May 28). *Open Source Software in the Department of Defense.* Retrieved 1/20/2008 from http://iase.disa.mil/policy-guidance/oss-in-dodmemo.pdf.

U.S. Department of Defense, Chairman of the Joint Chiefs of Staff Instruction (CJCSI) (2006, Mar 8). CJCSI 6212.01, *Interoperability and Supportability of Information Technology and National Security Systems.* Retrieved 4/10/2008 from http://www.dtic.mil/cjcs_directives/cdata/unlimit/6212_01.pdf.

U.S. Department of Defense (2002). *Information Assurance.* (U.S. Department of Defense Directive 8500.01). Retrieved 2/18/2008 from www.dtic.mil/whs/directives/corres/pdf/851001p.pdf.

U.S. Department of Defense (2004). *Public Key Infrastructure and Public Key Enabling.* (U.S. Department of Defense Instruction 8520.2). Retrieved 2/18/2008 from http://www.dtic.mil/whs/directives/corres/pdf/852002p.pdf.

U.S. Department of Defense (2008). *Defense Federal Acquisition Regulation Supplement (DFARS), Subpart 207.1 – Acquisition Plans.* Retrieved 5/12/2008 from http://www.acq.osd.mil/dpap/dars/dfars/html/current/207_1.htm#207.106.

U.S. Department of Defense (2008). *Defense Federal Acquisition Regulation Supplement (DFARS); Technical Data Rights (DFARS Case 2006-D055).* Retrieved 5/12/2008 from http://www.acq.osd.mil/dpap/dars/dfars/changenotice/2007/20070906/E7-17422.htm.

U.S. Department of the Navy (2007). *Department of the Navy Open Source Software Guidance.* [Department of the Navy Chief Information Officer (DoN CIO) Memorandum]. Retrieved 1/22/2008 from www.doncio.navy.mil/Download.aspx?AttachID=261.

U.S. Deputy Under Secretary of Defense, Advanced Systems and Concepts (2006, Jun 7). *Open Technology Development Roadmap.* Retrieved 3/4/2008 from http://www.acq.osd.mil/jctd/articles/OTDRoadmapFinal.pdf.

U.S. Government Accountability Office (GAO) (2006, Mar). *Information Assurance – National Partnership Offers Benefits, but Faces Considerable Challenges.* Retrieved 2/27/2008 from http://www.gao.gov/new.items/d06392.pdf.

U.S. Navy Program Executive Office, Integrated Warfare Systems (PEO-IWS7) (2006, Jul 7). *Naval Open Architecture Contract Guidebook.* Retrieved 3/4/2008 from https://acc.dau.mil/CommunityBrowser.aspx?id=18016&view=w.

U.S. Office of Management and Budget (U.S. OMB) (2004, Jul 1). *Software Acquisition.* Retrieved 2/27/2008 from http://www.whitehouse.gov/omb/memoranda/fy04/m04-16.html.

Van Oers, M. (2007, Mar 20). *OSX Malware Not Taking Off Yet.* Retrieved 2/4/2008 from http://www.avertlabs.com/research/blog/index.php/2007/03/20/osx-malware-not-taking-off-yet/.

Wait, P. (2005, Jan 10). *Services Demand Security in Enterprise Deals.* Retrieved 1/10/2008 from http://www.gcn.com/print/24_1/31468-1.html.

Wheeler, David A. (2007, June). *Open Source Software (OSS) in U.S. Government Acquisitions.* Retrieved 3/4/2008 from https://www.softwaretechnews.com/stn_view.php?stn_id=42&article_id=83.

Wheeler, David A. (2007, April 16). *Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!* Retrieved 5/12/2008 from http://www.dwheeler.com/oss_fs_why.html.

Wolfe, A. (2004, Apr 9). *Green Hills Calls Linux 'Insecure' for Defense.* Retrieved 2/8/2008 from http://eetimes.com/showArticle.jhtml?articleID=18900949.

Yasin, R. (2007, Apr 2). *The Long Road Toward Standard Configuration.* Retrieved 1/15/2008 from http://www.gcn.com/print/26_07/43383-1.html.

Yasin, R. (2007, Jun 4). *Winged Migration.* Retrieved 1/15/2008 from http://www.gcn.com/print/26_13/44414-1.html?topic=workflow.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. COMMON ACCESS CARD TESTING

A. TEST ENVIRONMENT

This survey was conducted using various Intel IA-32 (x86) -based computers: one Dell Pentium IV desktop, one Dell Pentium III laptop, and one IBM Pentium-M laptop. All three systems were running the Red Hat Enterprise Linux 5.1 Desktop with all standard installation options except SE Linux, in order to simplify troubleshooting. The RHEL systems were all registered with the Red Hat Network (RHN) under the Naval Postgraduate School's (NPS) site license, allowing the simplified installation of those software packages available on the RHN.

RHEL clients use the yum (Yellowdog Updater, Modified) program to locate and download pre-compiled binary software packages from Red Hat's repositories. A graphical front-end called Pirut makes it even easier to search or browse the catalog and select available programs. However, systems registered with the RHN are only subscribed to a base software distribution channel by default, which limits the number of programs to which they have access. To allow the clients to “see” the larger pool of software in Red Hat's online repositories, an administrator must log into the RHN and add additional sub-channels to the profile of each computer (or group of computers). In this study, the RHEL Desktop Supplementary and RHEL Desktop Workstation channels were enabled for all three machines.

Over the past several years, the SCM Microsystems SCR331 USB has become one of the most commonly-used CAC readers throughout the DoD. It is a white plastic external device, meant to sit on a user's desktop. Older versions of the reader may contain firmware that is not CCID 1.0 compatible. Also, some variants sold by ActivIdentity may look physically similar, but unfortunately use a different firmware which is not compatible with the USB CCID driver. This has been addressed in other documents, including

the “CAC on a Mac” literature produced at NPS (http://stinet.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA445103) and again in OSSG's documentation. To summarize these previous findings, the older SCM readers and ActivCard (now ActivIdentity) USB 2.0 readers can be “upgraded” so that they contain the latest firmware of the SCM Microsystems devices, effectively turning them into capable SCR331's. That procedure was followed for this study. One noteworthy stumbling block, however, was the version of ActivCard Gold middleware used on the Windows platform doing the upgrade. The newer version 6.x, which is being distributed by the U.S. Air Force for home use, would not perform the firmware flash; the result was consistently a complete system crash (blue screen). When this middleware was removed and replaced with the older ActivCard 3.x, the flash process worked.

The Dell SK3106 keyboard and newer Dell smart card keyboards (in use at computer labs around NPS) were also used in this study and worked without any modification.

B. FIREFOX WEB BROWSER

Firefox is the default web browser bundled with RHEL 5, and there are two ways of making it operate with CAC's. The first (and simplest) method is to use Network Security Services (NSS) files provided by DISA's Open Source Steering Group (OSSG):

1. From a computer with a NIPRNet connection (address that resolves to .mil), go to http://ossg.disa.mil/projects/linuxcac/ and download the RPM (Red Hat Package Manager) files containing the DoD CA certificates.
2. Close Firefox.
3. Install the RPMs by double-clicking them.
4. Go to your home folder, and click on *View*, then *Show Hidden Files*.
5. Browse to the ./mozilla/firefox directory. There, you will see one more folder made of a random-looking series of characters; this is the profile directory. Open that folder.

6. Backup (change the names of) cert8.db and secmod.db files, as they will be replaced.
7. Copy the cert8.db and secmod.db files from /etc/pki/nssdb (provided by OSSG's RPMs) to the Firefox profile directory where the originals were found in the previous step.

All of the DoD CA's are now loaded into Firefox's Certificate Authorities list (courtesy of the cert8.db file), and a new PKCS#11 security module has been loaded as a security device (from the secmod.db file). Upon visiting a CAC-enabled website, you should be prompted for your “master password”, which is your PIN.

Alternatively, the PKCS#11 module can be manually added as a security device from within Firefox:

1. Under the Preferences, Advanced menu, Security tab, click on Security Devices, then the Load button.
2. Choose any name for the Module Name, and specify the path to the Coolkey library (in RHEL5, use: /usr/lib/pkcs11/libcoolkeypk11.so).

Upon restarting the browser with the CAC inserted in the reader, the token will be accessible to the system, but will not typically be used unless all required DoD Certificate Authorities (CA) have been manually imported as well. One method is to export individual CA certificate files from Internet Explorer on a Windows client, then import them one by one into Firefox. This is a very tedious process, especially compared with the simplicity of copying the cert8.db file provided by OSSG.

As an added benefit, OpenOffice.org automatically takes advantage of NSS secmod.db and cert8.db files that have been loaded in the user's Firefox profile. Therefore, after the steps above have been completed, OpenOffice.org can also use the CAC to digitally sign documents (though only if they are saved in the OpenDocument .odt format).

C. EVOLUTION E-MAIL CLIENT

A similar procedure must be followed to gain CAC functionality in Evolution. First, install the OSSG CA RPMs as described above.

1. From a computer with a NIPRNet connection (address that resolves to .mil), go to http://ossg.disa.mil/projects/linuxcac/ and download the RPM files containing the CA certificates.
2. Go to your home folder and click *View*, then *Show Hidden Files*.
3. Browse to the .evolution directory.
4. Backup (change the names of) cert8.db and secmod.db files, as they will be replaced.
5. Copy the cert8.db and secmod.db files from /etc/pki/nssdb to the ~/.evolution directory (in step 3).

Again, there is a manual install method (provided by a Navy Research Labs website: (https://airborne.nrl.navy.mil/PKI), but this is not recommended since one must still import all the CA certificates manually. From a command prompt in your home directory, simply type:

modutil -add "Coolkey" -libfile /usr/lib/pkcs11/libcoolkeypk11.so -dbdir .evolution

Once the CAC is working in Evolution, it is possible to sign emails immediately, but one must still obtain the public keys of recipients in order to send encrypted emails. For this purpose, DISA's Global Directory Service Query (https://dod411.gds.disa.mil/) is invaluable. Simply search for a DoD user, then download their public key and import it into Evolution by going to *Edit, Preferences, Certificates*, the *Contact Certificates* tab, and clicking *Import*. Be sure to have your CAC inserted before visiting the GDS Query site.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Douglas E. Brinkley
   Naval Postgraduate School
   Monterey, California

4. Rex Buddenberg
   Naval Postgraduate School
   Monterey, California

5. Donna Burych
   Naval Postgraduate School
   Monterey, California

6. Diana Petross
   Naval Postgraduate School
   Monterey, California

7. Karl Pfieffer
   Naval Postgraduate School
   Monterey, California

8. Daniel Warren
   Naval Postgraduate School
   Monterey, California

9. Dan C. Boger, PhD
   Naval Postgraduate School
   Monterey, California